ANALYSIS OF SOFTWARE RELIABILITY

DISSERTATION

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR
THE AWARD OF THE DEGREE OF

Master of Philosophy

IN

STATISTICS

By

SHASHI SAXENA

Under the Supervision of

Dr. ARIF-UL-ISLAM

DEPARTMENT OF STATISTICS AND OPERATIONS RESEARCH
ALIGARH MUSLIM UNIVERSITY
ALIGARH (INDIA)

2008

DEPARTMENT OF
STATISTICS& OPERATIONS RESEARCH
ALIGARH MUSLIM UNIVERSITY,
ALIGARH - 202002, INDIA.

Certificate

This is to certify that ***Ms. Shashi Saxena*** *has carried out the work reported in the present dissertation entitled* ***"Analysis of Software Reliability"*** *under my supervision. The dissertation is her own work and I recommend it for consideration for the award of* ***Master of Philosophy*** *in* ***Statistics.***

Dated : 30/7/08

Dr. Arif-Ul-Islam
arifislam2@yahoo.com
(Supervisor)

Dedicated to

My

Parents

And Teachers

ACKNOWLEDGEMENT

I want to thank to 'GOD', the Almighty, the most powerful and the merciful whose blessings enabled me to complete this work in the present form.

I feel great pleasure at the completion of this work and would like to show my sincere gratitude to all the special persons who helped me.

It is, of course, my privilege and pleasure to express my profound sense of gratitude to my supervisor, ***Dr. Arif-Ul-Islam,*** *Department of Statistics and Operations Research, A.M.U., Aligarh for not only being dedicated and preserving but also his guidance, which seldom do justice to the latent qualities inherent in him. I must appreciate him for his patience, great involvement and sympathetic behavior, which enabled me to complete this work within stipulated time.*

I would like to thank ***Prof. A. H. Khan, Chairman, Department of Statistics and Operations Research A.M.U.,*** *Aligarh for his concern.*

I would like to thank to my teachers Prof. M. Z. Khan, Prof. M. J. Ahsan, Prof. M. M. Khalid, Dr. A. Bari, Dr. M. Yaqub, Dr. R. U. Khan and Dr. Haseeb Athar.

I would like to thank to all non-teaching staff members of the department, for their kind help and cooperation.

I am greatly indebted to many of my research colleagues and friends who have encouraged me throughout this work. I am particularly grateful to Ms. Shazia Zarrin, Mr. Devendra Kumar, Mr. Rahul Varshney,

Ms. Trapti Thakur, Ms. Yojna and Ms. Neeru for their constant encouragement and sharing of ideas during the preparation of manuscript.

My heart goes out in reverence to my parents and family members for their tremendous patience, forbearance, endurance and affection. All my appreciation for their support will not be adequate and enough to match their good wishes.

Shashi Saxena

Department of Stats and O.R.

sshashisaxena@gmail.com

Date: 30|07|08.

CONTENTS

PREFACE

Starting in the early 1959s, the word reliability acquired a highly specialized technical meaning in relation to the control of quality of manufactured product. As per the official definition of the Electronics Industries Association (EIA), quoted in "Reliability Principles & Practices" by S. R. Calabro, the reliability is, "the Probability of a device performing its purpose adequately for the period of time intended under the operating conditions encountered". The interest in Reliability theory currently exhibited by Engineers, mathematicians, economists, industrial managers and those concerned with the environmental and life sciences has stimulated the research work in this field. Electrical, Electronic and Mechanical equipments are being increasingly used in a number of fields- in industries for control of processes, in computers, in Medical Electronics, Atomic Energy, Communications, navigation at sea and in the air and many other fields. It is essential that an equipment should operate reliably under all conditions in which it is used. However, the more reliable a device is, the more difficult it is to measure its reliability. This is so because many years of testing under actual operating conditions would be required to obtain numerical measures of its reliability. Even if such testing was feasible, the rate of technical advance is so great that parts would be obsolete by the time their reliability had been measured. In addition, many of the components used in practice are subjected environments that are difficult to stimulate in the laboratory.

The present dissertation is devoted to the study of **"Analysis of Software Reliability"**. The subject matter of the dissertation has been arranged in five chapters. First chapter entitled **"Basic Concepts of Reliability Theory"** is of an introductory nature and we discuss the concept of

reliability, some important statistical probability distributions with properties and accelerated life test with models.

Second chapter entitled **"Basics of software Reliability"** deals with their software reliability and its importance. We discuss life cycle models with their advantages and drawbacks, software testing methodology and software reliability cost models.

Third chapter entitled **"Change-Point Problems in Software and Hardware Reliability"** deals with the problem of change-point. Change-point problems are generalized by studying the multi-path change-point where several independent sequences are considered simultaneously and each sequence has one change-point.

Fourth chapter entitled **"Inference for the Software Reliability Using Asymmetric Loss Functions: A Hierarchical Bayes Approach"** deals with the estimation of unknown number of errors in a piece of computer software.

Fifth chapter entitled **"Parameter Estimation of Some NHPP Software Reliability Models with Change-Point"** deals with Software Reliability growth models and the estimation of parameters.

References of the books and journals consulted through the task are given at the end of the dissertation.

CHAPTER 1

BASIC CONCEPTS OF RELIABILITY THEORY

1.1 INTRODUCTION

RELIABILITY

We often talk of an 'object' being reliable in the sense that it can be trusted to perform a certain job to the satisfaction of the 'USER' under 'normal conditions'. For example, a car is said to be reliable if we are sure to complete our journey without any breakdown on the way, provided nothing unusual (like hailstorm, fog, torrential rain or an accident) happens. Of human beings, newspersons often talk of 'reliable sources'. In both the cases the word reliable means 'dependable' or 'trustworthy'.

The scientific meaning of the term reliability is 'repeatability' or 'consistency'. A measure is considered reliable if it would give us the same result over and over again (assuming that whatever we are measuring is not changing).

Reliability as a concept in Industrial Engineering can be defined as 'freedom from failure', 'the ability to perform the specified mission' for a specified time under specified conditions.

In the field of Statistics, the reliability is defined as the characteristics of an item expressed by the probability that it will perform a required function under stated conditions for a stated period of time.

MEASUREMENT OF RELIABILITY

Out of several definitions available, the most comprehensive definition of reliability is given by Crowder et al. (1991):

"Reliability of a system (or a component) refers to its ability to operate properly according to a specified standard".

Going by this definition, it is felt that different measures of reliability are necessary, as different devices may have different objectives and standard. The use of a certain device actually determines the kind of reliability measures that is most meaningful and most useful. For example, the reliability measure associated with nuclear power reactor components is frequently taken to be the failure rate, since failure of a reactor is of primary concern. On the other hand, a power supply for a deep space probe must function without failure for the entire mission duration and so the probability of survival for the mission, is the most important measure of reliability. We now describe a commonly used measure of reliability that is based on the probability of an item that functions until first failure, functioning beyond some specified time.

Reliability Function: *Reliability is described by the reliability function R(t) that is the probability that a system or a component will carry out its mission through time t (Rigdon and Basu (2000))*

The reliability function (also called the survival function) evaluated at time t is just the probability that the failure time T is beyond time t. Thus, the relation that defines the reliability function is given by

$$R(t) = P(T \geq t) = 1 - F(t), \tag{1.1.1}$$

where $F(t)$ is the cumulative distribution function of the failure time T, which is supposed to be a random variable.

$$F(t) = \int_0^t f(t)dt \tag{1.1.2}$$

The cumulative distribution function is also known as unreliability function, and is represented by *Q (t).*

$$Q(t) = F(t) = \int_0^t f(t)dt \tag{1.1.3}$$

These two states are also mutually exclusive. Since reliability and unreliability are the probabilities of these two mutually exclusive states, the sum of these probabilities is always equal to unity. So then:

$$Q(t) + R(t) = 1$$

$$R(t) = 1 - Q(t)$$

$$R(t) = 1 - \int_0^t f(t)dt$$

$$R(t) = \int_t^\infty f(t)dt \tag{1.1.4}$$

1.2 BASIC CONCEPTS OF RELIABILITY

The Expected Life (Mean Time To Failure):

The mean time to failure (MTTF) is expected time during which the component will perform successfully and is defined as:

$$E(T) = \int_0^\infty t\, f(t)dt \tag{1.2.1}$$

where $f(t)$ is the pdf of T, the lifetime of the item. As the lifetime of an item has to be non-negative, we must have $f(t)$ defined for $T \geq 0$. Another convenient method for determining the expected life is given by:

$$E(T) = \int_0^\infty R(t)dt \tag{1.2.2}$$

This may be shown to be true by integrating by parts. $E(t)$ is also known as mean time to failure (MTTF).

Failure Rate And Hazard Function:

The failure process is usually quite complex and it is often difficult to understand the mechanics of the underlying process. It is even more difficult to describe a failure process mathematically. However, these difficulties can be overcome by applying the concept that permits different distributions to be distinguished on the basis of physical considerations. Such a concept is expressed as a hazard rate. A closely related concept is that of failure rate.

Failure Rate:

The rate at which the failures occur in a certain time interval $[t_1,t_2]$ is called the failure rate during that interval. It is defined as the probability that a failure per unit time occurs in the interval, given that a failure has not occurred prior to time t_1, the beginning of the interval. Thus the failure rate is given by:

$$\lambda(t) = \frac{\int_{t_1}^{t_2} f(t)dt}{(t_2 - t_1)\int_{t_1}^{\infty} f(t)dt}$$

$$= \frac{\int_{t_1}^{\infty} f(t)dt - \int_{t_2}^{\infty} f(t)dt}{(t_2 - t_1)\int_{t_1}^{\infty} f(t)dt} \qquad (1.2.3)$$

If we substitute $t_1 = t$ and $t_2 = t + \Delta t$, we get

$$\lambda(t) = \frac{R(t) - R(t+\Delta t)}{\Delta t R(t)} \tag{1.2.4}$$

Note that the failure rate is a function of time period.

The rate in the above definition is expressed as failure per unit time. In practice, the time unit might be replaced by kilometers, revolutions, stress and so on.

Hazard Rate:

The hazard rate (or hazard rate function or simply hazard function) is defined as the limit of the failure rate as the length of the interval $[t_1, t_2]$ approaches zero. Thus, it is instantaneous failure rate.

The hazard rate $h(t)$ is defined as:

$$\begin{aligned} h(t) &= \lim_{\Delta t \to 0} \frac{R(t) - R(t+\Delta t)}{\Delta t R(t)} \\ &= \frac{1}{R(t)}\left[-\frac{d}{dt}R(t)\right] \\ &= -\frac{d \ln R(t)}{dt} \\ &= \frac{f(t)}{R(t)} \end{aligned} \tag{1.2.5}$$

The quantity $h(t)dt$ represents the probability that a device of age t will fail in the small interval of time t to $t+\Delta t$. The importance of the hazard rate is that it indicates the change in the failure rate over the life span of the device. For example, two designs may provide the same reliability at a specified point in time; however the failure rates up to this point in time may differ. The failure rate is analogous to the death rate, in actuarial theory, as the hazard function is analogous to the force of mortality.

A typical Hazard rate generally has the so-called bathtub shape shown in fig.1.2.1

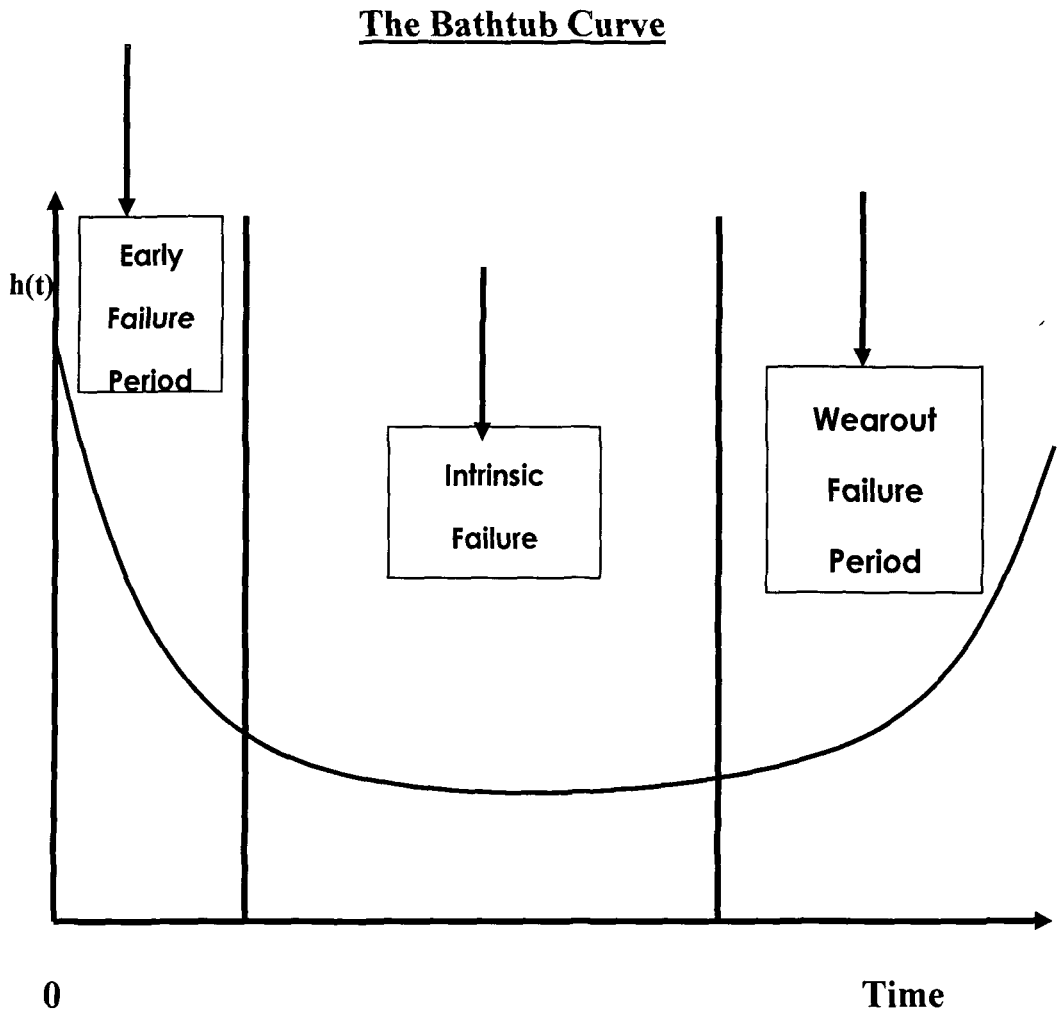


Fig 1.2.1: A typical (bathtub) hazard rate curve

In the above figure three distinct failure regions are indicated. The first, called the initial failure region, is characterized by a decreasing failure rate. It represents early failures due to material or manufacturing defects. Good quality control and burn in product testing may reduce the chance of early failure or even eliminate it altogether.

The second region, called the chance or random failure region, is characterized by a constant failure rate. It represents chance failures caused by sudden stresses, unusually severe and unpredictable operating

conditions and so on. To minimize or eliminate these would require a device that is over designed for the vast majority of situations.

The third position, called the wear-out failure region, is typified by an increasing failure rate, resulting from equipment deterioration, accumulated shocks, fatigue etc.

Thus it may be more convenient to select a distribution of the shape characteristic of the hazard rate rather than the shape of the pdf.

It can be shown mathematically that a hazard function must satisfy the condition

$$\int_0^\infty h(t)dt = \infty \qquad (1.2.6)$$

where $h(t) \geq 0$ for all $t \geq 0$.

Cumulative Hazard Function:

Based on the concept of hazard function, we also define Cumulative Hazard Function or Integrated Hazard Function given by:

$$H(t) = \int_0^t h(\tau)d\tau, \qquad t \geq 0 \qquad (1.2.7)$$

It is easy to see that cumulative hazard function satisfies the following:

(i) $H(0) = 0$

(ii) $\lim_{t\to\infty} H(t) = \infty$

(iii) $H(t)$ is non decreasing.

1.3 SOME IMPORTANT PROBABILITY DISTRIBUTIONS

A statistical distribution is fully described by its pdf (or probability density function). We use the definition of the pdf to show how all other functions most commonly used in reliability engineering and life data analysis can be derived. The reliability function, failure rate function,

mean time function and median life function can be determined directly from the pdf. We discuss some important distributions and their important features and characteristics.

1.3.1 EXPONENTIAL DISTRIBUTION

The Exponential distribution is a very commonly used distribution in Reliability (Engineering) Statistics just as the Normal distribution in other areas of statistics. Due to its simplicity, it has been very widely employed even in the cases where its use may not be convincingly justified. Davis (1952), Epstein (1958), Barlow and Proschan (1965) are among those who have put forth arguments in its favor.

The exponential distribution is inherently associated with the Poisson Process. Exponential distribution also occurs in several other contexts, such as the waiting time problems. Maguire, Pearson and Wynn (1952) studied mine accidents and showed that time intervals between accidents follow Exponential distribution.

The single parameter-exponential distribution is given by:

$$f(t) = \lambda e^{-\lambda t} = \frac{1}{m} e^{-T/m} \quad T \geq 0, \lambda > 0, m > 0 \qquad (1.3.1.1)$$

where,

λ = Constant failure rate, in failure per unit of measurement, e.g. failure per hour, per cycle etc.

m = mean time between failures or to a failure.

T = operating time, life or age (in hours, cycles, miles, etc).

This distribution requires the knowledge of only one parameter, λ, for its application.

CHARACTERISTICS OF EXPONENTIAL DISTRIBUTION

The Mean Time To Failure (MTTF):

The mean, $\overline{T}$ or mean time to failure (MTTF) of the one-parameter Exponential distribution is given by:

$$\overline{T} = \int_0^\infty t.f(t)dt = \int_0^\infty t.\lambda.e^{-\lambda t}dt = \frac{1}{\lambda}$$

The Median:

The median, $\breve{T}$ of the one-parameter Exponential distribution is given by:

$$\breve{T} = \frac{1}{\lambda}0.693$$

The Mode:

The mode, $\tilde{T}$ of the one-parameter Exponential distribution is given by

$$\tilde{T} = 0$$

The Standard Deviation:

The standard deviation, σ_T, of one-parameter Exponential distribution is given by:

$$\sigma_T = \frac{1}{\lambda} = m$$

The Reliability Function:

The one-parameter Exponential reliability function is given:

$$R(T) = e^{-\lambda t} = e^{-T/m}$$

This function is the complement of the Exponential cumulative distribution function or,

$$R(T) = 1 - Q(T) = 1 - \int_0^T f(T)dT$$

and, $$R(T) = 1 - \int_0^T \lambda e^{-\lambda T} dT = e^{-\lambda T}$$

The Conditional Reliability Function:

The Exponential conditional reliability equation gives the reliability for a mission of t duration, having already successfully accumulated T hours of operation upto the start of this new mission. The Exponential conditional reliability function is

$$R(T \mid t) = \frac{R(T+t)}{R(T)} = \frac{e^{-\lambda(T+t)}}{e^{-\lambda t}} = e^{-\lambda t}$$

which says that the reliability for a mission of t duration undertaken after the component or equipment has already accumulated T hours of operation from age zero is only a function of mission duration and not a function of the age at the beginning of the mission. This is referred as the memory less property.

The Exponential Reliable Life:

The reliable life or the mission duration for a desired reliability goal t_R for the one-parameter Exponential distribution is given by:

$$R(t_R) = e^{-\lambda t_R}$$

$$\ln[R(t_R)] = -\lambda t_R$$

$$t_R = \frac{\ln[R(t_R)]}{\lambda}$$

The Exponential Failure Rate Function:

The Exponential failure rate function is given by:

$$\lambda(T) = \frac{f(T)}{R(T)} = \frac{\lambda e^{-\lambda(T)}}{e^{-\lambda(T)}} = \lambda = \text{Constant}$$

The characteristics of one-parameter Exponential distribution can be exemplified by examining its parameter, λ , and the effect that λ has on the *pdf* , reliability and failure rate functions.

EFFECT OF λ ON THE *pdf*

- The scale parameter is $\frac{1}{\lambda}$
- As λ is decreased in value, the distribution is stretched out to the right, and as λ is increased, the distribution is pushed towards the origin.
- The distribution has no shape parameter as it has only one shape i.e. the exponential. The only parameter, it has is, the failure rate, λ.
- The distribution starts at $T = 0$, the level of $f(T) = \lambda$ and decreases thereafter exponentially and monotonically as T increases and is convex.
- As $T \to \infty$, $f(T) \to 0$.
- This *pdf* can be thought of as a special case of Weibull *pdf* with $\beta = 1$.

Effect Of λ On the Reliability Function:

- The one-parameter Exponential reliability function starts at the value of $1-F(T)$ at $T=0$. It decreases thereafter monotonically and is convex.
- As $T \to \infty$, $R(T \to \infty) \to 0$.

Effect Of λ On the Failure Rate Function:

The failure rate function for the Exponential distribution is constant and it is equal to the parameter λ.

1.3.2 WEIBULL DISTRIBUTION

Of all the probability distributions available for reliability problems, weibull distribution is the most commonly used probability distribution in the field of industrial engineering as well as for failure data analysis (also known as life data analysis).

The distribution is named after Waloddi Weibull, a Swedish physicist, who used it in 1939 to represent the distribution of the breaking strength of materials. Kao, J.H.K.(1958-1959) advocated the use of this distribution in reliability studies and quality control work. Leiblein and Zelen (1956) used it as a model for ball bearing failures. Mann (1968) gave a variety of situations in which the distribution is used for other types of failure data.

The two-parameter Weibull *pdf* is given by:

$$f(T)=\frac{\beta}{\eta}\left(\frac{T}{\eta}\right)^{\beta-1} e^{-\left(\frac{T}{\eta}\right)^{\beta}} \qquad (1.3.2.1)$$

where,

$$f(T)\geq 0,\ T\geq 0,\ \beta>0,\ \eta>0$$

and,

η=Scale parameter

β=Shape parameter or slope.

CHARACTERISTICS OF THE WEIBULL DISTRIBUTION

The Mean Time to Failure (MTTF):

The mean, $\overline{T}$ of the two-parameter Weibull *pdf* is given by:

$$\overline{T} = \eta\Gamma\left(\frac{1}{\beta}+1\right)$$

where $\Gamma\left(\frac{1}{\beta}+1\right)$ is the gamma function evaluated at the value of $\left(\frac{1}{\beta}+1\right)$.

The Median:

The median, $\breve{T}$ of the two-parameter Weibull *pdf* is given by:

$$\breve{T} = \eta(\ln 2)^{\frac{1}{\beta}}$$

The Mode:

The mode, $\tilde{T}$ of the two-parameter Weibull *pdf* is given by:

$$\tilde{T} = \eta\left(1-\frac{1}{\beta}\right)^{\frac{1}{\beta}}$$

The Standard Deviation:

The standard deviation, σ_T of the two-parameter Weibull *pdf* is given by:

$$\sigma_T = \eta\sqrt{\Gamma\left(\frac{2}{\beta}+1\right)-\Gamma\left(\frac{1}{\beta}+1\right)^2}$$

The *cdf* and the Reliability Function:

The *cdf* of the two-parameter Weibull *pdf* is given by:

$$F(T) = 1 - e^{-\left(\frac{T}{\beta}\right)^{\beta}}$$

The Weibull Reliability Function is given by:

$$R(T) = 1 - F(T) = e^{-\left(\frac{T}{\eta}\right)^{\beta}}$$

The Conditional Reliability Function:

The Weibull conditional reliability function is given by:

$$R(T \mid t) = \frac{R(T+t)}{R(T)} = \frac{e^{-\left(\frac{T+t}{\eta}\right)^{\beta}}}{e^{-\left(\frac{T}{\eta}\right)^{\beta}}}$$

$$R(T \mid t) = e^{-\left[\left(\frac{T+t}{\eta}\right)^{\beta}-\left(\frac{T}{\eta}\right)^{\beta}\right]}$$

Above equation gives the reliability for a new mission of t duration, having already accumulated T hours of operation upto the start of this new mission and the units are checked out to assure that they will start the next mission successfully. (It is called conditional because you can calculate the reliability of a new mission based on the fact that the unit(s) already accumulated T hours of operation successfully).

The Reliable Life:

For the two-parameter Weibull distribution, the reliable life, T_R, of a unit for a specified reliability, starting the mission at age zero, is given by:

$$T_R = \eta\{-\ln[R(T_R)]\}^{\frac{1}{\beta}}$$

This is the life for which the unit will function successfully with a reliability of $R(T_R)$. If $R(T_R) = 0.5$ then $T_R = \tilde{T}$, the median life, or the life by which half of the units will survive.

The Failure Rate Function:

The two-parameter Weibull failure rate function $\lambda(T)$ is given by:

$$\lambda(T) = \frac{f(T)}{R(T)} = \frac{\beta}{\eta}\left(\frac{T}{\eta}\right)^{\beta-1}$$

The characteristics of the two-parameter Weibull distribution can be exemplified by examining the two parameters, β and η, and the effect they have on the *pdf*, reliability and failure rate functions.

EFFECTS OF β ON THE *pdf*:

- For $0 < \beta < 1$, the failure rate decreases with time.
- As $T \to 0, f(T) \to \infty$.
- As $T \to \infty, f(T) \to 0$
- $F(T)$ decreases monotonically and is convex as T increases.
- The mode is non-existent.
- For $\beta = 1$, it becomes the exponential distribution, as a special case

Or,

$$f(T)=\frac{1}{\eta}e^{-\frac{T}{\eta}}, \qquad \eta>0, T\geq 0$$

where $\frac{1}{\eta}=\lambda=$ chance, useful life or failure rate.

- For $\beta>1, f(T)$, the weibull assumes wear out type shapes (i.e. the failure rate increases with time).
- $f(T)=0$ at $T=0$
- $f(T)$ increases as $T\rightarrow\tilde{T}$ (mode) and decreases thereafter.
- For $\beta=2$ it becomes the Rayleigh distribution as a special case. For $\beta<2.6$ the Weibull *pdf* is positively skewed (has a right tail), for $2.6<\beta<3.7$ its coefficient of skewness approaches zero (no tail); consequently, it may approximate the normal *pdf* and for $\beta>3.7$ it is negatively skewed (left tail).
- The parameter β is a pure number i.e. it is dimensionless.

Effects Of β On Reliability Function:

- $R(T)$ decreases sharply and monotonically for $0<\beta<1$, it is convex and decreases less sharply for the same β.
- For $\beta=1$ and the same β, $R(T)$ decreases monotonically but less sharply than for $0<\beta<1$ and is convex.
- For $\beta>1, R(T)$ decreases as T increases but less sharply than before and as wear-out sets in, it decreases sharply and goes through an inflection point.

Effects Of β On Failure Rate Function:

The Weibull failure rate for $0<\beta<1$ is unbounded at $T=0$. The failure rate $\lambda(T)$, decreases thereafter monotonically and is convex, approaching the value of zero as $T\to\infty$ or $\lambda(\infty)=0$. This behavior makes it suitable for representing the failure rate of units exhibiting early-type failures, for which the failure rate decreases with age. When such behavior is encountered, the following conclusions can be drawn:

- Burn-in testing and/or environmental stress screening are not well implemented.
- There are problems in the production line.
- Inadequate quality control.
- Packaging and transit problem.
- For $\beta=1$, $\lambda(T)$ yields a constant value of $\frac{1}{\eta}$, or,

$$\lambda(T)=\lambda=\frac{1}{\eta}$$

- For $\beta>1$, $\lambda(T)$ increases as T increases and becomes suitable for representing the failure rates of units exhibiting wear-out type failures. For $1<\beta<2$ the $\lambda(T)$ curve is concave, consequently the failure rate increases at a decreasing rate as T increases.
- For $\beta=2$, or for the Rayleigh distribution case, the failure rate function is given by

$$\lambda(T)=\frac{2}{\eta}\left(\frac{T}{\eta}\right)$$

hence there emerges a straight-line relationship between $\lambda(T)$ and T, starting at a value of $\lambda(T)=0$ at $T=0$ and increasing thereafter with a slope of $\frac{2}{\eta^2}$. Consequently, the failure rate increases at a constant rate as T increases. Furthermore, if $\eta=1$ the slope becomes equal to 2 and $\lambda(T)$ becomes a straight line that passes through the origin with a slope of 2.

- When $\beta>2$ the $\lambda(T)$ curve is convex, with its slope increasing as T increases. Consequently, the failure rate at an increasing rate as T increases indicating wear-out life.

- A change in the scale parameter η has the same effect on the distribution as a change of the abscissa scale.

- If η is increased, β is kept the same, the distribution gets stretched out to the right and its height decreases, while maintaining its shape and location.

- If η is decreased, while β is kept the same, the distribution gets pushed towards the left (i.e. towards its beginning, or 0) and its height increases.

1.4 ACCELERATED LIFE TESTS

Many devices such as electronic items have very high reliability when operating within there intended normal use environment. This presents problem in measuring the reliability of such devices because a very long period of testing under the actual operating conditions would be required to obtain sufficient data to estimate the reliability. Even if this testing could be accomplished, the time frame is such that the devices may become absolute before their reliability is established due to the high rate of technological advances. Also, it would be difficult to conduct the testing in laboratory.

One solution to the problem of obtaining meaningful life test data for high reliability devices is accelerated life testing. This type of testing involves observing the performance of these kinds of devices operating at higher stress levels than usual to obtain failures more quickly. In order to shorten product life, it is a well established engineering practice to use certain stresses or accelerating variables, such as higher levels of temperature, voltage, pressure, vibration, etc., than the normal operating level.

The main difficulty of accelerated life testing lies in using the failure data obtained at the accelerated, or higher stress, condition to predict the reliability, mean life, or other quantities under the normal use condition. Extrapolation from the accelerated stresses to the normal use stress is done

by choosing an appropriate model, called an ***Acceleration Model***. The choice of an acceleration model calls for a knowledge of the variation of failure behavior with environment. In parametric methods, this involves functional relationship between the parameters of the failure distributions and the environmental stresses. The relationship may also involve

unknown parameters. In non-parametric approaches, where no specific form of the failure distribution is specified, the change in the failure distribution due to a change in environmental stress is assumed. In either the parametric or non-parametric all unknown parameters must be estimated from the accelerated test data in order to extrapolate to the normal use stress.

Four acceleration models are used, i.e. Power Rule model, the Arrhenius model, the Eyring model, and the Generalized Eyring model. These models will be discussed by Mann, Schafer, and Singhpurwall (1974).

1.4.1 ACCELERATION MODELS

The use of acceleration life testing to make inferences about the normal use life distribution requires a model to relate the life length to the stress levels that are to be applied to items being tested. This model is referred to as the acceleration model.

Here some acceleration models that have been used in parametric and non-parametric method will be described briefly.

In parametric, suppose the life time random variable X_i^0 of items in an environment described by a constant stress level V_i has a probability distribution $F^0(t;\underline{\theta}_i)$ depending on a vector of parameter $\underline{\theta}_i$. Two assumptions which are made (Mann, Schafer, and Singpurwalla, 1974) are

(i) The change in stress level does not change the type of the lifetime distribution $F^0(t;\underline{\theta})$, but changes only the parameter values.

(ii) The relationship between the stress level V and the parameters $\underline{\theta}$, say $\underline{\theta}=m(V;\alpha,\beta...)$, is known except for one or more of the acceleration parameter $\alpha;\beta...$ and that the relationship is valid for a certain range of

the elements of V. The objective here is to obtain estimates of the parameters $\alpha;\beta\ldots$ based on life test data obtained at large values of V and makes inferences about $\underline{\theta}$ for the normal use stress V_0.

The exponential distribution with parameter λ is widely used as a lifetime distribution. So the acceleration models will be discussed here for exponential distributions. Several authors have considered other lifetime distributions such as Weibull [Mann (1972), and Nelsen (1975)], extreme value [Meeker and Nelson (1975), and Nelsen and Meeker (1978)], and lognormal [Nelson and Kielpinsiki (1976)]. Suppose that under constant application of single stress at level V_i, the item being tested has an exponential lifetime distribution with mean μ_i given by

$$f^o(t;\lambda_i) = \lambda_i \exp(-\lambda_i t), t \geq 0, i > 0$$

$$= 0, otherwise.$$

Then $\mu_i = 1/\lambda_i$ is the mean time to failure under stress level V_i. The following acceleration models (relationships between λ_i and V_i) have been suggested in the literature.

1.4.2.1 THE POWER RULE (OR INVERSE POWER) MODEL

This model can be derived by considerations of kinetic theory and activation energy. This model has applications to fatigue testing of metals, the dielectric breakdown of capacitors, and aging of multi component systems. The model is:

$$\mu_i = \alpha V_i^{-\beta},\ \alpha > 0, \beta > 0$$

and this implies that the mean time of failure μ, decreases as the β^{th} power of the applied voltage V. it is desirable to estimate α and β from

life test data at stress levels $V_1,\ldots,V_K$ and make inferences about $\mu_0 = 1/\lambda_0$ at the normal use stress V_0.

1.4.1.2 THE ARRHENIUS MODEL

This model expresses the degradation rate of a parameter of the device as the function of its operating temperature. It is usually applied to thermal aging and is applicable to semiconductor materials. Here

$$\lambda_i = Exp(\alpha - \beta / V_i)$$

is the model, where V_i denotes the temperature stress and α and β are unknown parameters to be estimated in order to make inferences about λ_0 at normal temperature level V_0.

1.4.1.3 THE EYRING MODEL FOR A SINGLE STRESS

This model can be derived from principles of quantum mechanics and its expresses the time rate of degradation of some device parameter as a

function of the operating temperature. Here

$$\lambda_i = V_i \exp(\alpha - \beta / V_i)$$

is the model.

1.4.1.4 THE GENERALIZED EYRING MODEL

This model has application to accelerated testing of devices subjected to a constant application of two types of stresses, one thermal and one non-thermal. The model is

$$\lambda_i = \alpha T_i \exp(-\beta / KT_i)\exp(\gamma^{V_i} + \delta V_i / KT_i)$$

where α, β and γ are unknown parameters to be estimated, K denotes Boltzmann's constant, whose value is $1.38*10^{-16}$ erg / degree Kelvin, and T_i is the thermal stress level and V_i is the non-thermal stress. In the absence of a non-thermal stress, this model reduces to

$$\lambda_i = \alpha T_i \exp(-\beta / T_i).$$

CHAPTER 2

BASICS OF SOFTWARE RELIABILITY

2.1 INTRODUCTION

Software now controls banking systems, all forms of telecommunications, process control in nuclear plants and factories as well as defense systems. Even in households without a PC, many of the gadgets and the automobiles are software controlled. The society has developed an extraordinary dependence on software. There are many well-known cases of tragic consequences of software failures. In popular software packages used everyday, a very high degree of reliability is needed because the enormous investment of the software developer is at stake. Studies have shown that reliability is regarded as the most important attribute by potential customers.

It is not possible to write software which is totally defect free, except possibly for very small programs. All programs must be tested and debugged until sufficiently high reliability is achieved. Total elimination of all faults in large software systems is infeasible. Software must be released at some point in time as further delay will cause unacceptable loss of revenue and market share. The developer must take a calculated risk and must have a strategy for achieving the required reliability by the target release date.

In recent past, enough data has available to develop and evaluate methods for achieving high reliability. Developing reliable software has become an engineering discipline rather than an art. For hardware systems, quantitative methods for achieving and measuring reliability have been in

universal use for a long time. Similar techniques for software are coming in use due to emergence of well-understood and validated approaches.

Here we will use the terms *failure* and a *defect* as defined below.

Failure – A department of the system behavior from user requirements during execution.

Defect (or fault) – An error in system implementation that can cause a failure during execution.

A defect will cause a failure only when the erroneous code is executed and the effect is propagated to the output. The *testability* of the defect is defined as the probability of detecting it with a randomly chosen input. Defects with very low testability can be very difficult to detect.

Some mathematical concepts are applicable to both software and hardware reliability. Hardware faults often occur due to aging. Combined with manufacturing variation in the quality of identical hardware components, the reliability variation can be characterized as exponential decay with time. On the other hand, the software reliability improves during testing as bugs are found and removed. Once released, the software reliability is fixed. The software will fail time to time during operational use when it cannot respond correctly to an input. Reliability of hardware components is often estimated by collecting failure data for a large number of identical units. For a software system, its own past behavior is often a good indicator of its reliability, even though data from other similar software systems can be used for projections.

2.2 DEVELOPMENT PHASES

A competitive and mature software development organization targets a high reliability objective from the very beginning of software

development. Generally, the software life cycle is divided into the following phases:

A. Requirements and definitions: In this phase the developing organization interacts with the customer organization to specify the software system to be built.

B. Design: In this phase, the system is specified as an interconnection of units, such that each unit is well defined and can be developed and tested independently.

C. Coding: In this phase, the actual program for each unit is written, generally in a high level language such as C or C^{++}.

D. Testing: This phase is a critical part of the quest for high reliability and can take 30 to 60% of the entire development time. It is generally divided into these separate phases:

1. **Unit test:** In this phase, each unit is separately tested and changes are done to remove the defects found.
2. **Integration testing:** During integration, the units are gradually assembled and subsystems are tested partially.
3. **System testing:** The system as a whole is exercised during system testing.
4. **Acceptance testing:** The purpose of this test phase is to assess the system reliability and performance in the operational environment.
5. **Operational use:** Once the software developer has determined that an appropriate reliability criterion is satisfied, the software is released.
6. **Regression testing:** When significant additions or modification are made to an existing version, regression testing is done on the new or 'build' versions to ensure that it still works and has not 'regressed' to lower reliability.

Table 1: Defects introduced and found during different phases:

	Defect (%)		
Phase	Introduced	Found	Remaining
Requirements analysis	10	5	5
Design	35	15	25
Coding	45	30	40
Unit test	5	25	20
Integration test	2	12	10
System test	1	10	1

Life Cycle Models-

Many different software life cycles have been proposed. These have different motivations, strengths and weaknesses. The life cycle models generally require the same type of tasks to be carried out, the only difference is the ordering of the tasks in time.

Different software development process (or life-cycle) models:

- Waterfall model
- Rapid prototyping
- Evolutionary development
- Component reuse
- V model
- Formal transformation

WATERFALL MODEL:

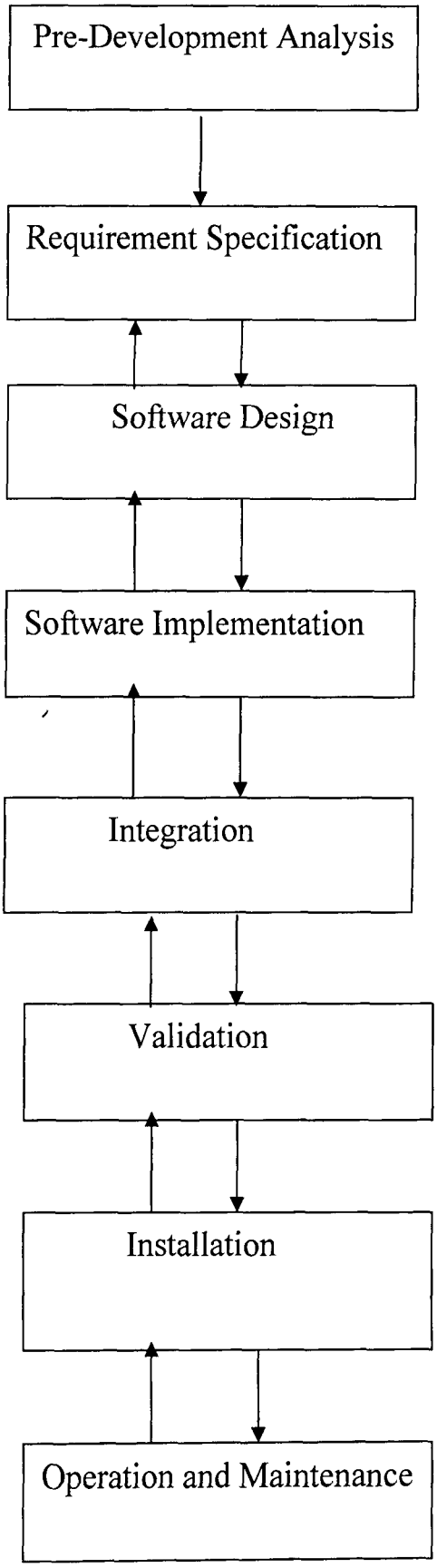


Fig. 2.1: Waterfall Lifecycle Model

Advantages:

- Each phase must be completed before the next starts.
- Original modal did not allow iteration.
- Still most widely used in industry and standards.

Drawbacks:

- Does not support parallel activity.
- No working software until very late.
- Does not support reuse.
- Does not support customer involvement.

2.3 SOFTWARE RELIABILITY AND ITS IMPORTANCE

Software reliability is one of the important parameters of software quality and system dependability. It is defined as the probability of failure-free software operation in a specified environment for a specified period of time [Lyu 96]. It means that the probability that given software operates failure free for a specified time on the machine for which it was designed, given that it was within design limits and that the last failure occurred at a given time. A software failure occurs when the behavior of the software departs from its specifications and it is the result of a software fault or a design defect, being activated by certain input to the code during its execution.

The issue of designing reliable software has acquired its importance due to the following reasons:

- Systems are becoming software intensive.
- Many software intensive systems are safety critical.
- Software users are demanding reliable, warranted software systems.
- The cost of software development is increasing.

2.4 SOFTWARE RELIABILITY MEASURES

The classical reliability theory generally deals with hardware. In hardware system the reliability decays because of the possibility of permanent failures. However, this is not applicable for software. During testing, the software reliability grows due to debugging and becomes constant once defect removal is stopped. The following are the most commonly used reliability measures.

Durational reliability: Following classical reliability terminology, we can define *reliability* of software system as:

$$R(T) = P\{\text{no system failure during}(0,t)\} \tag{2.4.1}$$

Transaction reliability: Sometimes a *single-transaction* reliability measure, as defined below, is more convenient to use.

$$R = P\{\text{a single transaction will not encounter a failure}\} \tag{2.4.2}$$

Both measures above assume *normal operation*, i.e. the input mix encountered obeys the operational profile (defined below):

Mean Time To Failure (MTTF): The expected duration between two successive failures.

Failure intensity (λ): The expected number of failures per unit.

Note that:

$$MTTF = \frac{1}{\lambda} \tag{2.4.3}$$

Since testing attempts to achieve a high defect-finding rate, failure intensity during testing λ_t is significantly higher than λ_{op}, failure intensity during operation. Test-acceleration factor A is given by:

$$A = \frac{\lambda_t}{\lambda_{0p}} \tag{2.4.4}$$

and is controlled by the test selection strategy and the type of application.

2.5 SOFTWARE TESTING METHODOLOGY:

To test a program, a number of inputs are applied and the program response is observed. If the response is different from the expected, the program has at least one defect. Testing can have two separate objectives. During debugging, the aim is to increase the reliability as fast as possible, by finding faults as quickly as possible. On the other hand, during certification, the object is to assess the reliability, thus the fault finding rate should be representative of actual operation. The test generation approaches can be divided into the classes.

2.5.1 BLACK-BOX (OR FUNCTIONAL) TESTING:

The black box approach is a testing method in which test data are derived from the specified functional requirements without regard to the final program structure. It is also termed as data-driven, input/output driven or requirement-based testing. Because only the functionality of the software module is of concern, black-box testing also mainly refers to functional testing. A testing method emphasized on executing the functions and examination of their input and output data. The tester treats the software under test as a black box. Only the inputs, outputs and specifications are visible and the functionality is determined by observing the outputs to corresponding inputs. In testing, various inputs are exercised and the outputs are compared against specification to validate the correctness. All test cases are derived from the specification. No implementation details of the code are considered. It is obvious that the more we have covered in the input space, the more problems we will find and therefore we will be more confident about the quality of the software. Ideally we would be tempted to exhaustively test the input space. But as stated above, exhaustively testing the combinations of valid inputs will be impossible for most of the programs. Therefore, we should consider invalid inputs,

timing and resource variables separately. Combinatorial explosion is the major roadblock in functional testing. To make things worse, we can never be sure whether the specification is either correct or complete. Due to limitations of the language used in the specifications (usually natural language), ambiguity is often inevitable. Even if we use some type of formal or restricted language, we may still fail to write down all the possible cases in the specification. Sometimes, the specification itself becomes an intractable problem. It is not possible to specify precisely every situation that can be encountered using limited words and people can seldom specify clearly what they want. They usually can tell whether a prototype is (or is not) what they want after they have been finished. Specification problems contribute approximately 30% of all bugs in software.

The research in black-box testing mainly focuses on how to maximize the effectiveness of testing with minimum cost, usually the number of test cases. It is not possible to exhaust the input space, but it is possible to exhaustively test a subset of the input space. Partitioning is one of the common techniques. If we have partitioned the input space and assume all the input values in a partition is equivalent then we only need to test one representative value in each partition to sufficiently cover the whole input space. Domain testing partitions the input domain into regions and considers the input values in each domain an equivalent class. Domains can be exhaustively tested and covered by selecting a representative value(s) in each domain. Boundary values are of special interest. Experience shows that test cases that explore boundary conditions have a higher payoff than test cases that do not. Boundary value analysis requires one or more boundary values selected as representative test cases. The difficulties in domain testing are that incorrect domain definitions in the specifications cannot be efficiently discovered.

Good partitioning requires knowledge of the software structure. A good testing plan will not only contain black-box testing but also white-box approaches and combination of the two.

2.5.2 WHITE-BOX (OR STRUCTURAL) TESTING:

Contrary to black-box testing, software is viewed as a white box or glass-box in white-box testing, as the structure and flow of the software under test are visible to the tester. Testing plans are made according to the details of the software implementation such as programming language, logic and styles. Test cases are derived from the program structure. White-box testing is also called glass-box testing, logic-driven testing or design-based testing.

There are many techniques available in white-box testing because the problem of intractability is eased by specific knowledge and attention on the structure of the software under test. The intention of exhausting some aspect of the software is still strong in white-box testing and some degree of exhaustion can be achieved such as executing each line of code at least once (statement coverage), traverse every branch statement (branch coverage) or cover all the possible combinations of true and false condition predicates (multiple condition coverage).

Control-flow testing, loop testing and data flow testing, all maps the corresponding flow structure of the software into a directed graph. Test cases are carefully selected based on the criterion that all the nodes or paths are covered or traversed at least once. By doing so we may discover unnecessary 'dead code' (code that is of no use or never gets executed at all) which cannot be discovered by functional testing.

In mutation testing, the original program code is perturbed and many mutated programs are created, each contains one fault. Each faulty version of the program is called the mutant. Test data are selected based

on the effectiveness of failing the mutants. The more mutants a test case can kill, the better the test case is considered. The problem with mutation testing is that it is too computationally expensive to use. The boundary between black-box approach and white-box approach is not clear-cut. Many testing strategies mentioned above, may not be safely classified into black-box testing or white-box testing. It is also true for transaction-flow testing, syntax testing, finite-state testing and many other testing strategies not discussed in this text. One reason is that all the other techniques will need some knowledge of the specification of the software under test. Another reason is that the idea of specification itself is broad. It may contain any requirement including the structure, programming language and programming style as part of the specification content.

We may be reluctant to consider random testing as a testing technique. The test case selection is simple and straightforward: they are randomly chosen. Study indicates that random testing is more cost effective for many programs. Some very subtle errors can be discovered with very low cost and it is also not inferior in coverage than other carefully designed testing techniques. One can also obtain reliability estimate using random testing results based on operational profiles. Effectively combining random testing with other testing techniques may yield more powerful and cost effective testing strategies.

2.6 SOFTWARE COST MODELS:

In defining important software cost factors, a cost model should help software and managers answer the following questions:

1. How should resources be scheduled to ensure the on-time and efficient delivery of a software product?
2. Is the software product sufficiently reliable for release (e.g. have we done enough testing)?

3. What information does a manager or software developer need to determine the release of software from current software testing activities?

The following notations and basic assumptions are applied throughout this chapter.

NOTATIONS

$m(T)$	expected number of errors to be detected by time T.
a	total number of software errors to be eventually detected.
b	exponential index.
$\lambda(T)$	fault detection rate per unit time or intensity function.
	mission time
$R(x/T)$	reliability function of software by time T for a mission time x.
T	software release time.
C_1	software test cost per unit time.
C_2	cost of removing each error per unit time during testing.
$E(T)$	expected total cost of a software system by time T.
y	time to remove an error during testing phase.
μ_y	Expected time to remove an error during testing phase which is $E(Y)$.

GENERAL ASSUMPTIONS

1. The cost to perform testing is proportional to the testing time.
2. The cost to remove errors during the testing phase is proportional to the total time of removing all the errors detected by the end of testing phase.
3. There is a risk cost related to the reliability at each release time point.

4. The time to remove error during testing follows a truncated exponential distribution.
5. Without loss of generality, the Goel-Okumoto NHPP model will be used as a reliability function.

Let Y be a random variable of time to remove an error. Based on assumption (4), the probability density function of Y is given by

$$s(y) = \frac{\lambda e^{-\lambda y}}{\int_0^{T_0} \lambda e^{-\lambda z} dz}, \quad 0 \le y \ge T_0 \tag{2.6.1}$$

Where T_0 is the maximum time to remove an error. The expected time to remove each error is

$$\mu_y = E(y) = \int_0^{T_0} y s(y) dy = \int_0^{T_0} \frac{y\lambda e^{-\lambda y}}{\int_0^{T_0} \lambda e^{-\lambda z} dz} dy \tag{2.6.2}$$

After simplification we obtain

$$\mu_y = \frac{1 - (\lambda T_0 + 1)e^{-\lambda T_0}}{\lambda(1 - e^{-\lambda T_0})} \tag{2.6.3}$$

2.6.1 A SOFTWARE COST MODEL WITH RISK FACTOR

The expected software system cost, $E(T)$, is defined as:

(1) The cost to perform testing;

(2) The cost incurred in removing errors during the testing phase; and

(3) A risk cost due to software failure.

A. The cost to perform testing is given by

$$E_1(T) = C_1(T)$$

B. The expected total time to remove all $N(T)$ errors is

$$E[\sum_{i=1}^{N(T)} Y_i] = E[N(T)]E[Y_i] = m(T)\mu_y$$

Hence the expected cost to remove all errors detected by time T can be expressed as

$$E_2(T) = C_2 E[\sum_{i=1}^{N(T)} Y_i] = C_2 m(T) \mu_y$$

C. The risk cost due to software failure after releasing the software is

$$E_3(T) = C_3[1 - R(x/T)]$$

where C_3 is the cost due to software failure.

Therefore, the expected total software cost can be expressed [Zhang, 1998] as:

$$E(T) = C_1 T + C_1 T m(T) \mu_y + C_3[1 - R(x/T)]$$

The mean value function $m(T)$ is

$$m(T) = a(1 - e^{-bt})$$

The error detection rate function is

$$\lambda(T) = abe^{-bt}$$

The reliability of the software is

$$R(x/T) = e^{-[m(T+x) - m(T)]}$$

$$= e^{-a[e^{-bt} - e^{-b(T+x)}]}$$

2.6.2 A GENERALIZED SOFTWARE COST MODEL:

Notations:

C_0 Set-up cost for software testing.

C_3 Cost of removing an error per unit time during the operational phase.

C_4 Loss due to software failure.

W Variable of time to remove an error during the warranty period in the operational phase.

μ_W Expected time to remove an error during the warranty period in the operational phase, which is $E(W)$.

T_W Period of warranty time.

α The discount rate of the testing cost.

Additional Assumptions:

(6) There is a set-up cost at the beginning of the software development process.

(7) The cost of testing is a power function of the testing time. This means that at the beginning of the testing, the cost increases with a higher gradient, slowing down later.

(8) The time to remove each error during the warranty period follows a truncated exponential distribution.

(9) The cost to remove errors during the warranty period is proportional to the total time of removing all errors detected between the intervals of (T, T_W).

Similarly, from assumption 8, the truncated exponential density function of error removal time during warranty period is

$$q(w) = \frac{\lambda_W e^{-\lambda_W w}}{\int_0^{T_0} \lambda_W e^{-\lambda_W w}} \qquad for\, 0 \le w \ge T_0$$

Therefore, the expected time to remove an error during the warranty period is

$$\mu_W = \frac{1-(\lambda_W T_0 + 1)e^{-\lambda_W T_0}}{\lambda_W (1 - e^{-\lambda_W T_0})}$$

The expected software system cost comprises of the set-up cost, the cost to do testing, the cost incurred in removing errors during the testing phase and during the warranty period and the risk cost in releasing the software

system by time T. Hence, the expected total software system cost $E(T)$ can be expressed as follows (Pham, (1999)):

$$E(T) = C_0 + C_1 T^{\alpha} + C_2 m(T)\mu_y + C_3[m(T+T_w) - m(T)]\mu_y + C_4[1 - R(x/T)]$$

where $0 \le \alpha \le 1$

2.6.3 A COST MODEL WITH MULTIPLE FAILURE ERRORS:

In this section, a software cost model is presented under the following assumptions:

a. The cost of debugging an error during the development phase is lower than in the operational phase.

b. The cost of removing a particular type of error is constant during the debugging phase.

c. The cost of removing a particular type of error is constant during the operational phase.

d. The cost of removing critical errors is more expensive than major errors and the cost of removing major errors is more expensive than minor errors.

e. There is a continuous cost incurred during the entire time of the debugging period.

Notations

T Software release time.

C_{i1} Cost of fixing a type i error during the test phase $i = 1,2,3$.

C_{i2} Cost of fixing a type i error during the operation phase.

$(C_{i2} \ge C_{i1}, i = 1,2,3)$

C_3 Cost of testing per unit time.

$E(T)$ Expected cost of software.

R_0 Pre-specified software reliability.

T_r Debugging time required to attain minimum cost subject to a reliability constraint.

T_e Debugging time required to attain minimum cost subject to the number of remaining error constraint.

T_{rel} Debugging time required to attain maximum reliability subject to a cost constraint.

$$E(T) = \int_0^T \left[C_3 t + \sum_{i=1}^{3} C_{i1} m_i(t) \right] .g(t)dt$$

$$+ \int_T^{\infty} \left[C_3 T + \sum_{i=1}^{3} C_{i1} m_i(T) + \sum_{i=1}^{3} C_{i2} (m_i(t) - m_i(T)) \right] .g(t)dt$$

2.6.4 COST SUBJECT TO RELIABILITY CONSTRAINT:

Consider the expected software cost $E(T)$ and the software reliability $R(x/T)$ as the evaluation criteria. We determine the optimum release time that minimizes the expected software cost subject to attain a desired reliability level, R_0. Then the optimization problem can be formulated as

Minimize $E(T)$

Subject to $R(x/T) \geq R_0$

where,

$$R(x/t) = \exp\left[-\left(\sum_{i=1}^{3} \frac{ap_i}{1-\beta_i} (e^{-(1-\beta_i)b_i})(1 - e^{-(1-\beta_i)b_i x}) \right) \right]$$

$$E(T) = \int_0^T \left[C_3 t + \sum_{i=1}^{3} C_{i1} m_i(t) \right] .g(t)dt$$

$$+ \int_T^{\infty} \left[C_3 T + \sum_{i=1}^{3} C_{i1} m_i(T) + \sum_{i=1}^{3} C_{i2} (m_i(t) - m_i(T)) \right] .g(t)dt$$

2.6.5 COST SUBJECT TO THE NUMBER OF REMAINING ERRORS CONSTRAINTS:

Consider both the expected total software system cost, $E(T)$, and the expected number of failure type I errors remaining in the system, $\overline{m}_i(T)$, as the evaluation criteria. The optimal release problem can be formulated as

$$Minimize\ E(T)$$

$$Subject\ to\ \overline{m}_i(T) \le d_i\ i = 1,2,3.$$

where

$$\overline{m}_i(T) = m_i(\infty) - m_i(T) = \frac{ap_i}{1-\beta_i} e^{-(1-\beta_i)b_i T}$$

and d_i is the accepted number of remaining typeI errors.

Define

$$T_{m_i} = \frac{\ln\left[\frac{ap_i}{d_i(1-\beta_i)}\right]}{(1-\beta_i)b_i}$$

The function $\overline{m}_i(T)$ is, of course, decreasing in T for all T. Then

$\overline{m}_i(T) \le d_i$ if and only if $T \ge T_{m_i}$.

2.6.6 SOFTWARE RELIABILITY SUBJECT TO COST CONSTRAINT

Consider both the software reliability $R(x/T)$ and the expected software cost $E(T)$ as the evaluation criteria. The optimal policies problem can be formulated as

$$Maximize\ R(x/T) Subject\ to\ E(T) \le C_R$$

where C_R is the maximum amount allowable

CHAPTER 3

CHANGE-POINT PROBLEMS IN SOFTWARE AND HARDWARE RELIABILITY

3.1 INTRODUCTION

Most of the work in change-point problems is based on the following assumptions:

1) $X_1, X_2 \ldots X_\tau, X_{\tau+1} X_{\tau+2} \ldots X_n$ are independent.
2) $X_1, X_2, \ldots, X_\tau$ are identically distributed with distribution function F and $X_{\tau+1}, X_{\tau+2}, \ldots, X_n$ are identically distributed with distribution function G.

F and G are known to belong to parametric families (e.g. normal, binomial) or otherwise known in functional form.

The parameter τ is the change-point which is considered unknown and is to be estimated from the data.

The change-point problems have been studied by many authors. Hinkley (1970) used maximum likelihood to estimate τ in the situation where F and G are from the same parametric family or F and G may be arbitrary known distributions. The nonparametric estimation of the change-point has been discussed by Carlstein (1988). Joseph and Wolfson (1992) generalized the change-point problems by studying the multi-path change-points where several independent sequences are considered simultaneously and each sequence has one change-point. The bootstrap and empirical Bayes methods are also suggested to estimate the change-points.

In this chapter, we extend the change-point problems from another point of view, especially motivated from software reliability modeling. Let F and G be two different life time distributions with density functions $f(t)$ and $g(t)$, $X_1, X_2 \ldots X_\tau, X_{\tau+1} X_{\tau+2} \ldots X_n$ be the interfailure times of the sequential failures in a lifetime testing. We assume that

Assumption I. There is a finite number N of items under testing, the parameter N may be unknown.

Assumption II. At the beginning, all of the items have the same lifetime distribution F. After τ failures are observed, the remaining $(N-\tau)$ items have the distribution G. The change-point τ is assumed unknown.

Assumption III. The sequence $(X_1, X_2 \ldots X_\tau)$ is statistically independent of the sequence $(X_{\tau+1}, X_{\tau+2}, \ldots, X_n)$.

Here, we need not to assume the independence between the variables in sequences $(X_1, X_2 \ldots X_\tau)$ and $(X_{\tau+1}, X_{\tau+2}, \ldots, X_n)$, because the interfailure times of failures in a lifetime testing are usually dependent.

Note that assumption Ⅲ may not be realistic in some cases. We use it only for the sake of the model simplicity. However, it is easy to modify this assumption or drop it.

In software reliability, the initial number of faults contained in a program is always unknown. One has to execute the program in a specific environment and improves its quality by detecting and correcting the faults. Many software reliability models assume that during the fault detection process, each failure caused by a fault occurs independently and randomly in time according to the same distribution, see e.g. Musa et al. (1987). The failure distribution can be affected by many factors such as the running environment, testing strategy and the resource. Generally, the running environment may not be homogeneous and can be changed

with the human learning process. Hence, the change-point problems are of interest in modeling the fault detection process. The change-point can be occurred when the testing strategy and resource allocation are changed. Also, the increasing knowledge of the program, the testing facilities and other random factors can be the causes of the change-points.

The change-point problems can also occur in hardware reliability and survival analysis. The difference is only that the parameter N is usually known in hardware reliability. For example, the change-point problem can occur in the lifetime testing in a random environment and when the testing equipments are replaced or repaired. Particularly, the observed data in this area are often grouped and are then dependent. For example, if we are investigating a new treatment for some disease, the observed data may be censored because some patients may be out of the trials and new patients may come into the trials. Hence, the obtained data are dependent unlike the common change-point models.

In the following sections, some change-point models are presented based on the assumptions I, II and III. Also, we consider the maximum likelihood estimations of the change-point τ and the other parameters such as the number N of items, the parameters of distribution F and G. The applications in software reliability and numerical example are given.

3.2 SOME SPECIFIC CHANGE-POINT MODELS

We further assume that a lifetime testing is performed according to the type-II censoring plan in which the number of observations n is decided before the data is collected.

Denote by $T_1, T_2, ..., T_n$ the arrival times of sequential failures in the lifetime testing. Then we have

$$T_1 = X_1$$

$$T_2 = X_1 + X_2$$

$$\vdots$$

$$T_n = X_1 + X_2 + \cdots + X_n \qquad (3.2.1)$$

According to the assumptions I, II and III the failure times $T_1, T_2, ..., T_\tau$ are the first τ order statistics of a sample with size N from parent distribution $F, T_{\tau+1}, T_{\tau+2}, \cdots, T_n$ are the first $(n-\tau)$ order statistics of a sample with size $(N-\tau)$ from parent distribution G.

3.2.1 *MODEL I* (JM model with one change-point):

Assume that F and G are Exponential distribution with parameters λ_1 and λ_2, respectively. Then, it is easy to show that the interfailure times $X_1, X_2, ..., X_n$ are independently exponentially distributed. Specifically, X_i is exponentially distributed with parameter $\lambda_1(N-i+1)$, $i = 1,2,\ldots,\tau$ and X_j is exponentially distributed with parameter $\lambda_2(N-\tau-j+1)$, $j = \tau+1, \tau+2, \ldots, n$.

The *JM model* was first proposed by Jelinski and Moranda (1972) to model the fault detection process of a program. It was derived based on the following assumptions:

$J1$. There are N initial faults in the program.

$J2$. A detected fault is removed instantaneously and no new fault is introduced.

$J3$. Each failure caused by a fault occurs independently and randomly in time according to an Exponential distribution.

The *JM model* is the simplest and most cited software reliability model. It is also the most criticized model because assumption $J2$ and $J3$ imply that all faults in a program have the same size and each removal of the detected faults reduces the failure intensity by the same amount. However, the *JM model* remains central to the topic of software reliability. Many models are derived based on the assumptions similar to that of *JM model*. Considering the change-point in this model is also intended to remove its drawbacks and the proposed model is expected to be more close to the reality.

3.2.2 *MODEL II* (Weibull Model):

Assume that F and G are Weibull distribution functions with parameters (λ_1, β_1) and (λ_2, β_2) respectively. That is

$$F(t) = 1 - \exp\{-\lambda_1 t^{\beta_1}\},$$

$$G(t) = 1 - \exp\{-\lambda_2 t^{\beta_2}\}.$$

In this case, the time intervals of failures are dependent. The *Weibull model* without change-points is used by Wagoner (1973) to describe the fault detection process. Particularly, when the shape parameter $\beta = 2$, the *Weibull model* is the model proposed by Schick and Wolverton (1973). In application, we can use the simplified model in which the shape parameters β_1 and β_2 are assumed equal.

3.2.3 *MODEL III* (Littlewood Model):

Assume that F and G are Pareto distribution functions given by

$$F(t) = 1 - (1 + \frac{t}{\lambda_1})^{\beta_1},$$

$$G(t) = 1 - (1 + \frac{t}{\lambda_2})^{\beta_2}.$$

When no change-point occurs, *model III* is the *Littlewood model* which is proposed by Littlewood (1981) based on assumption $J1, J2$ and $J3$, but the failure rates λ_i's associated with each fault are assumed to be identical and independent random variables with Gamma distribution. Under the Bayesian framework, the failure distribution is shown to be Pareto distribution, see Littlewood (1981) for details.

3.3 MAXIMUM LIKELIHOOD ESTIMATION:

In general, we assume that the distribution concerned belong to parameter families $\{F(t \mid \theta_1), \theta_1 \in \Theta_1\}$ and $\{G(t \mid \theta_2), \theta_2 \in \Theta_2\}$. Because $T_1, T_2, \ldots, T_\tau$ are the first τ order statistics of a sample with size N from parent distribution F, $T_{\tau+1}, T_{\tau+2}, \ldots, T_n$ are the first $(n-\tau)$ order statistics of a sample with size $(N-\tau)$ from parent distribution G, then the log likelihood without the constant term is

$$L(\tau, N, \theta_1, \theta_2 \mid T_1, T_2, \ldots, T_n)$$

$$= \sum_{i=1}^{n} (N - i + 1) + \sum_{i=1}^{\tau} f(T_i \mid \theta_1) + \sum_{i=\tau+1}^{n} g(T_i \mid \theta_2)$$

$$+ (N - \tau) \log\{1 - F(T_\tau \mid \theta_1)\} + (N - n) \log\{1 - G(T_n \mid \theta_2)\} \qquad (3.3.1)$$

The maximum likelihood estimator of the change-point is the value $\hat{\tau}$ that together with $(\hat{N},\hat{\theta}_1,\hat{\theta}_2)$ maximizes the expression (3.3.1). There is no closed form for $\hat{\tau}$. However, it can be obtained by calculating the log likelihood for each possible value of τ, $1 \le \tau \le n-1$, and selecting as $\hat{\tau}$ the value that maximizes the log likelihood.

For *model I,* the maximum likelihood estimates of parameter (N,λ_1,λ_2) given change-point τ in terms of the interfailure times are determined by the following equations:

$$\hat{\lambda}_1 = \frac{\tau}{\sum_{i=1}^{\tau}(\hat{N}-i+1)x_i} \tag{3.3.2}$$

$$\hat{\lambda}_2 = \frac{(n-\tau)}{\sum_{i=\tau+1}^{n}(\hat{N}-i+1)x_i} \tag{3.3.3}$$

$$\sum_{i=1}^{n}\frac{1}{(\hat{N}-i+1)} = \hat{\lambda}\sum_{i=1}^{\tau}x_i + \hat{\lambda}_2\sum_{i=\tau+1}^{n}x_i \tag{3.3.4}$$

On the distribution of $\hat{\tau}$, it is not easy to have a general conclusion. The asymptotic distribution of $\hat{\tau}$ may be deduced following Hinkely (1970). It is also possible to use the bootstrap to estimate the distribution of $\hat{\tau}$ following Joseph and Wolfson (1992). We should point out that there exists a problem when the parameter N is unknown. In some cases, maximum likelihood estimate of N doesn't exist, see e.g. Joe and Reid (1985), Wright and Hazelhurst (1988) and Huang (1990) where the *JM model* and other models are considered and the conditions for the

existence of maximum likelihood estimates of N are given. Consequently, when the maximum likelihood estimate of N doesn't exist for some data set, the maximum likelihood estimates of τ may not exist.

3.4 APPLICATION:

In this section, we apply *model I* for a real data set in software reliability. The data set is tabled in Table 3.1

First, we use the data in Table 3.1 to estimate parameters $(\tau, N, \lambda_1, \lambda_2)$ by formulae (3.3.2), (3.3.3) and (3.3.4). The estimates for parameters $(\tau, N, \lambda_1, \lambda_2)$ are

$$\hat{\tau} = 16,\ \hat{N} = 144.89,\ \hat{\lambda}_1 = 1.1086e-04,\ \hat{\lambda}_2 = 2.9925e-05$$

The estimates of parameters N and λ in the JM model are

$$\hat{N} = 141.90,\ \hat{\lambda} = 3.4967e-05.$$

We can see that the estimations for the number of faults N by these two models are slightly different.

For possible values of the change-point τ, the log likelihood is plotted in Figure 3.4.1.

Second, we consider the prediction ability of *model I*. Note that the main objective in software reliability is to predict the behavior of future failures as well as possible. Here, we compare *model I* with the *JM model* from the prediction point of view.

Table 3.1 Interfailure Times of the System T1 in Musa (1979)

3	30	113	81	115	9	2	91	112	15	138
50	77	24	108	88	670	120	26	114	325	55
242	68	422	180	10	1146	600	15	36	4	0
8	227	65	176	58	457	300	97	263	452	255
197	193	6	79	816	1351	148	21	233	134	357
193	336	31	369	748	0	232	330	365	1222	543
10	16	529	379	44	129	810	290	300	529	281
160	828	1011	445	296	1755	1064	1783	860	983	707
33	868	724	2323	2930	1461	843	12	261	1800	865
1435	30	143	108	0	3110	1247	943	700	875	245
729	1897	447	386	446	122	990	948	1082	22	75
482	5509	100	10	1071	371	790	6150	3321	1045	648
5485	1160	1864	4116							

The u-plot and prequential likelihood techniques are used to evaluate the prediction ability. In general, the u-plot of a good predicting system should be close to the diagonal line with unit slope and the maximum vertical distance between the u-plot and the diagonal line, which is called Kolmogorov distance, is a measurement of the closeness. On the other hand, it can be shown that *model A* is in favor of *model B* if the prequential likelihood ratio PL^A / PL^B is consistently increasing as the predicting process. For more details about these techniques, see Littlewood (1991) and Dawid (1998).

Figure 3.4.1: Values of the log likelihood function against the change-point

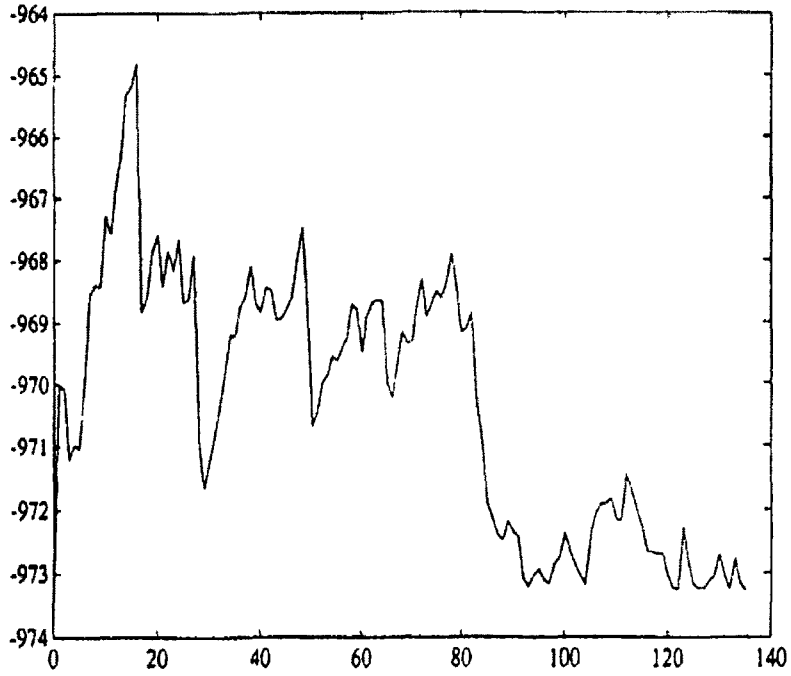


Figure 3.4.2: U-plots comparison between *Model I* and the JM model

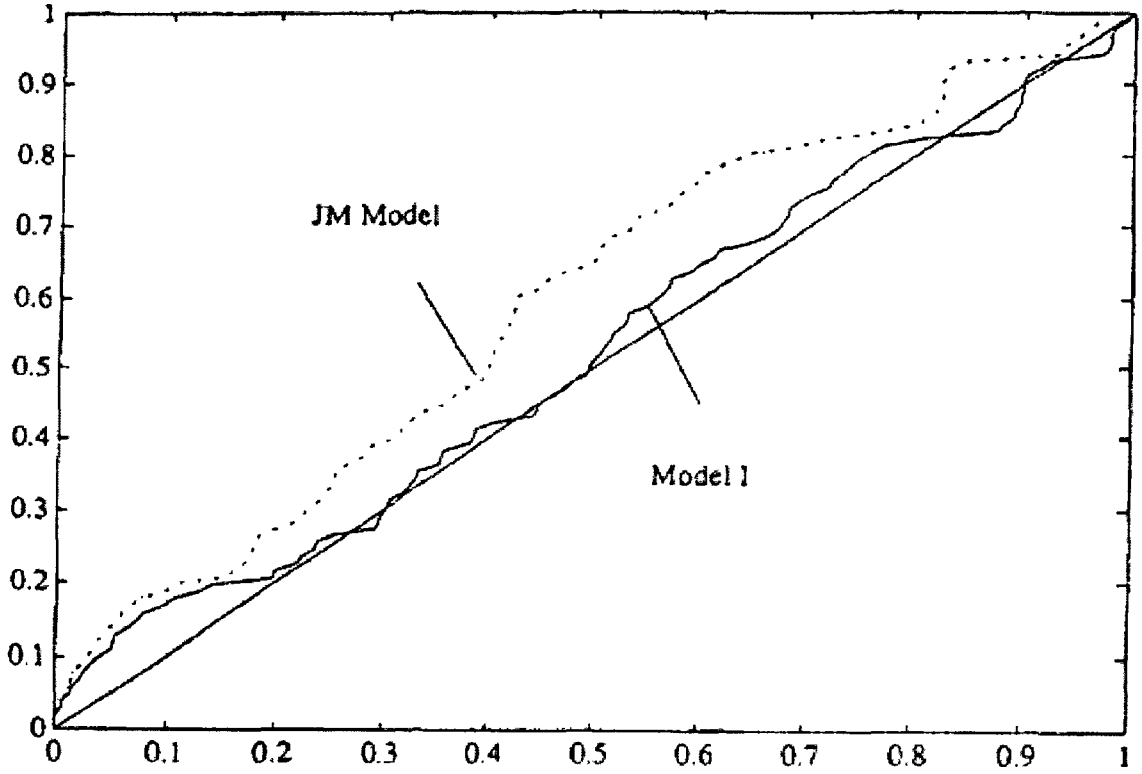
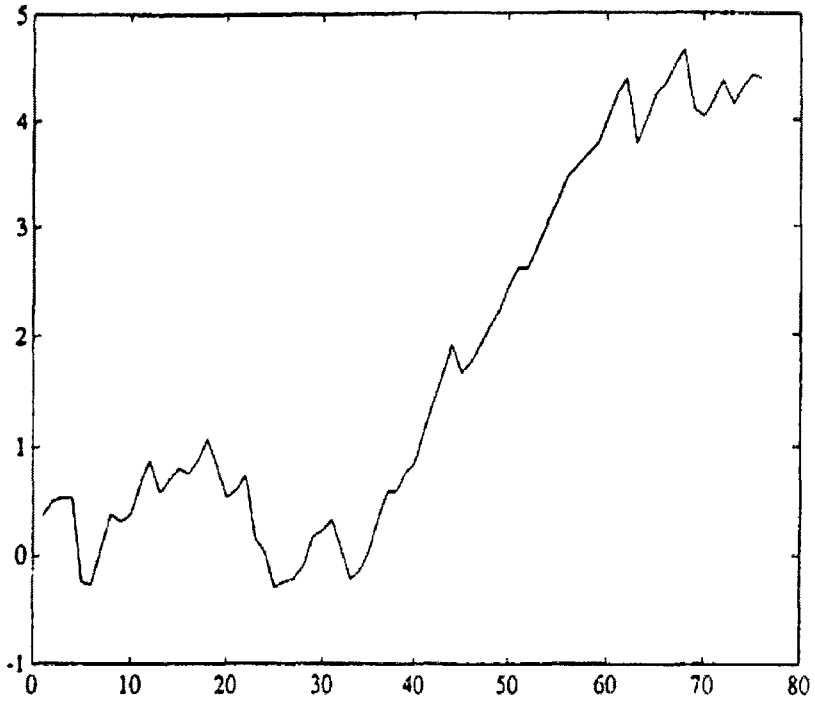


Figure 3.4.2 is the u-plot of *model I* and the JM model starting after 35 failures. The Kolmogorov distance for *model I* is equal to 0.088 and for the JM model it is 0.1874.

Figure 3.4.3: Log prequential likelihood ratios between *model I* and the *JM model*



From Figure 3.4.2 and 3.4.3, we can see that the fitness and the prediction have been improved by considering the change-point in *JM model* for the specific data set.

3.5 DICUSSION

We have presented some change-point models which can be used in software reliability. The main difference between the models proposed here and common ones is the dependent sample. However, there are numerous open questions about the change-point problems in software reliability.

The maximum likelihood estimation of the change-point is yet to be solved because in some software reliability models, the maximum likelihood estimation of the number of faults can be infinity. In such cases, another estimation procedure has to be developed.

An extension of the change-point problems in software reliability can be made by considering the multi-path change-points. It may be expected that the original software reliability models can be improved by introducing the idea of the change-points.

CHAPTER 3

CHANGE-POINT PROBLEMS IN SOFTWARE AND HARDWARE RELIABILITY

3.1 INTRODUCTION

Most of the work in change-point problems is based on the following assumptions:

1) $X_1, X_2 \ldots X_\tau, X_{\tau+1} X_{\tau+2} \ldots X_n$ are independent.
2) $X_1, X_2, \ldots, X_\tau$ are identically distributed with distribution function F and $X_{\tau+1}, X_{\tau+2}, \ldots, X_n$ are identically distributed with distribution function G.

F and G are known to belong to parametric families (e.g. normal, binomial) or otherwise known in functional form.

The parameter τ is the change-point which is considered unknown and is to be estimated from the data.

The change-point problems have been studied by many authors. Hinkley (1970) used maximum likelihood to estimate τ in the situation where F and G are from the same parametric family or F and G may be arbitrary known distributions. The nonparametric estimation of the change-point has been discussed by Carlstein (1988). Joseph and Wolfson (1992) generalized the change-point problems by studying the multi-path change-points where several independent sequences are considered simultaneously and each sequence has one change-point. The bootstrap and empirical Bayes methods are also suggested to estimate the change-points.

CHAPTER 4

INFERENCE FOR THE SOFTWARE RELIABILITY USING ASSYMETRIC LOSS FUNCTIONS: A HIERARCHICAL BAYES APPROACH

4.1 INTRODUCTION

Let N be the unknown number of errors in a piece of computer software. An important problem in software reliability is the estimation of N from past data.

The most commonly used model for describing the stochastic failures of software is that proposed by Jelinski and Moranda (1972) – henceforth J-M. The assumptions made by J-M are the following:

(i) When the software fails, the error causing the failure can be detected and removed without inserting any additional errors.

(ii) The failure rate of the software at any point is proportional to the residual number of faults in the program; the program begins with N faults.

(iii) Each of the N faults contributes an equal amount Λ (unknown) to the failure rate.

(iii) Given N and λ, the times between successive failures of the program, $T_1, T_2, \ldots, T_N$ are independent with the conditional density given by

$$f(t_i \mid N, \Lambda) = \Lambda(N - i + 1)\exp\{-\Lambda(N - i + 1)t_i\} \qquad (4.1.1)$$

with N and Λ being the unknown parameters.

There are some critical issues about the J-M model that are:

(i) The "bug counting" framework from which the model is derived is unrealistic (see Littlewood, 1981).

(ii) The maximum likelihood estimator (MLE) of N can often be misleading (see Forman and Singpurwalla (1977)).

(iii) The use of symmetric loss functions may be inappropriate to make inference for N, as has been recognized in the literature (see Zellner, 1986).

(iv) The stopping procedure for debugging the software proposed by Forman and Singpurwalla (1977) is empirical and needs a critical examination of the actual likelihood function at each stage of the procedure without considering an appropriate loss function.

(v) The squared loss function may be inappropriate for predicting future failure times.

In this chapter, we will discuss in details items (ii)-(iv) of the five critical issues of the J-M model that we presented above.

These and other matters are developed as follows:

Motivated by Raftery (1988), in section 4.2 we present a hierarchical Bayes approach to get information about N. Also, using the Table 4.1 introduced by Forman and Singpurwalla (1977) to study the instability of the MLE of N, a comparison of the posterior mode with the MLE is considered. In section 4.3, we suggest the use of LINEX loss function, which seems us to be more appropriate for estimating the parameter N and a stopping rule for debugging the software based on the robustness of this asymmetric loss function is formulated.

4.2 A HIERARCHICAL BAYES MODEL TO ESTIMATE N

Bayesian inference for the unknown parameters N and Λ involves assigning a prior distribution to the pair (N,Λ), and using the data and the Bayes theorem to obtain the posterior distribution. In this section we are concerned about the parameter N. A difficulty with the Bayesian analysis to estimate the parameter N has been the absence of a sufficiently flexible family of prior distribution, mainly due the fact that N is an integer. To avoid this problem, as in Raftery's paper (1988), we assume that N has a Poisson model distribution. Since the resulting hyperparameters are continuous –valued, the following hierarchical Bayes model can be formulated:

$$(i)\ T_i \mid N,\Lambda \sim \exp\{\Lambda(N-i+1)\} \qquad i=1,2,\ldots,n$$

$$(ii)\ N \mid \mu \sim Poisson(\mu)$$

$$(iii)\ \mu \sim Gamma[a_1,b_1],\Lambda \sim Gamma[c_1,d_1] \qquad (4.2.1)$$

Where μ and Λ are assumed to be independent. This model provides a kind of "unification" of many alternative models in software reliability because each alternative model is obtained by assigning particular prior distributions to the parameters N and Λ. We can show this "unification" by considering the following cases:

Case(i) Assume that μ and Λ in (iii) are degenerate at known values, then by the assumption A_2 and theorem 3.3 of Landeberg and Singpurwalla (1985) we have the Goel and Okumoto model (1979).

Case(ii) Assume that N and μ in (ii) and (iii) are degenerated at known values, then we have the Littlewood and Verral model (1973).

Case (iii) Assume that μ in (iii) is degenerated at a known value, then we has a combination of the Goel and Okumoto and the Littlewood and Verral models.

With the purpose of comparing our Bayes procedure with the MLE, we only consider the noninformative case, that is, we take $a_1 = b_1 = c_1 = d_1 = 0$. The advantage of the hierarchical Bayes model is in terms of understanding the model, the three-stage model is much easier to understand. Also, from (ii) and (iii), it is not difficult to see that N has a negative binomial distribution.

Let $t_1, t_2, \ldots, t_n$ be the observed times between failures, and let $t^{(n)} = (t_1, t_2, \ldots, t_n)$. Then given the observed data $t^{(n)}$, the likelihood functions for $N (N \geq n)$ and Λ is

$$L(N, \Lambda \mid t^{(n)}) = \Lambda^n \prod_{i=1}^{n} [N - i + 1] \exp\{-\Lambda s (N - a)\}, \qquad (4.2.3)$$

where $s = \sum_{i=1}^{n} t_i$ and $a = \sum_{i=1}^{n} (i-1) t_i / s$.

The joint posterior density of (N, Λ, μ) for $N \geq n$ is

$$p(N, \Lambda, \mu \mid t^{(n)}) \alpha \frac{e^{-\mu} \mu^{N-1}}{N!} \prod_{i=1}^{n} [N - i + 1] \Lambda^{n-1} \exp\{-\Lambda s (N - a)\} \qquad (4.2.4)$$

Integrating over μ and Λ, we obtain the marginal posterior density of N as

$$p(N \mid t^{(n)}) \alpha \frac{\prod_{i=1}^{n} [N - i + 1]}{N (N - a)^n}, \quad for\ N \geq n \qquad (4.2.5)$$

Now an important property of (4.2.5) is that it is improper. Not only is this posterior improper, but we certainly also have the property that the posterior mean does not exist. This problem can be avoided in two natural ways:

(a) Truncation:

Let N_0 such that (4.2.5) is close to zero. The improper posterior can be avoided by truncating $p(N|t^{(n)})$ away from n, that is, by making $p(N|t^{(n)})=0,\ N \geq N_0$.

(b) Informative prior on Λ:

By taking a gamma prior $\Gamma(c,d)$ on Λ and a noninformative prior on μ which, based on (ii) and (iii) of (4.2.2), is equivalent to taking the following improper noninformative prior on N:

$$p(N) \alpha 1/N$$

In this chapter we follow the first alternative.

If $n << N$, Forman and Singpurwalla (1977) showed via profile likelihood that the MLE of N may be very unstable with respect to a. They concluded that as a becomes large, i.e., when the times between failures in the latter stages of testing are greater than those during the earlier stages, the MLE of N tends to n. If a is small, the MLE of N tends to become large, i.e. MLE of N is unstable for small values of a. This situation could provide estimates of N that are highly misleading.

Let us denote the posterior mode of N by $\hat{N}_{MOD}$ and the MLE of N by $\hat{N}_{MLE}$ In Table 4.1, taking $n=5$, we compare the behavior of $\hat{N}_{MOD}$ and $\hat{N}_{MLE}$

Table 4.1 demonstrates very clearly that for small values of a, the Bayes estimator $\hat{N}_{MOD}$ is very stable when compared with the estimator $\hat{N}_{MLE}$.

Using the profile likelihood, Forman and Singpurwalla (1977) showed for $a = 0.52$ and $n = 2$ that the MLE of N, $\hat{N}_{MLE}$ is not finite. However, using (4.2.5), it is not hard to show that the posterior mode is finite, i.e. $\hat{N}_{MOD} = 2$

4.3 BAYES ESTIMATES UNDER ASSYMMETRIC LOSS FUNCTIONS

In the estimation of N the use of symmetric loss function may be inappropriate as has been recognized in the literature-see, for example, Zellner, (1986).

Table 4.1: Dependency of $\hat{N}_{MOD}$ and $\hat{N}_{MLE}$ on a

a	$n=5$	
	$\hat{N}_{MLE}$	$\hat{N}_{MOD}$
4.0	5.0	5.0
2.8	5.07	5.0
2.3	8.83	5.50
2.2	12.1	5.70
2.06	35.4	6.02
2.03	60.7	6.14
2.02	102.0	6.14
2,005	401.0	6.14
2,002	1,005.0	6.14
2,001	1,962.0	6.18
2,000	35,288.0	6.18

It seems to us that underestimation of the parameter N results in more serious consequence than overestimation of N. To take into account this problem, we suggest in this chapter the use of LINEX function (Zellner, 1986) which is formulated as follows:

$$L(\Delta) = b[\exp\{c\Delta\} - c\Delta - 1], \quad c > 0, b > 0 \tag{4.3.1}$$

where $\Delta = N - \hat{N}$. It is seen that, for $c = 1$, the function is quite asymmetric with underestimation being more costly than overestimation. If $c > 0$ and $\Delta > 0$, the loss grows exponentially as Δ grows whereas when $\Delta < 0$, the loss grows approximate linearly. Thus positive and negative errors of equal magnitude give rise to quite different losses when the asymmetric loss function (4.3.1) is employed. For small values of c, the function is almost symmetric and not far from a squared loss function (see Varian (1975) for more details). It is not difficult to see that Bayes estimator that minimizes the posterior expectation of the LINEX loss function with respect to $p(N \mid t^{(n)})$ is given by:

$$\hat{N}_{LINEX} = -\frac{1}{c}\ln\{\sum_{n<N\le N_0} e^{-cN} p(N \mid t^{(n)})\} \quad c > 0 \tag{4.3.2}$$

Also, it can be demonstrated that:

1. $E(N \mid t^{(n)}) \ge \hat{N}_{LINEX}$

2. $\hat{N}_{LINEX}$ is a decreasing function of c $(c > 0)$. (4.3.3)

For the standard debugging procedure, Forman and Singpurwalla (1977) proposed an empirical stopping rule based upon the relative likelihood function for the data, generated by simulation. Here motivated by (4.3.3) we propose the following sequential stopping rule for debugging the software:

1. Given that n errors are detected and $c = c_0$, compute

$$d(n.c_0) = E(N \mid t^{(n)}) - \hat{N}_{LINEX}$$

2. If $d(n,c_0)$ is close to zero, stop testing and accept the software and use the posterior mean or the optimal Bayes estimator under the LINEX function to estimate the parameter N. If not, observe another failure time t_{n+1} and go to step 1.

This stopping rule provides a kind of robustness with respect to the loss functions, i.e., we do not need to be worried about symmetric and asymmetric loss functions to estimate the parameter N . Also, it seems that this procedure is more realistic than the stopping rule suggested by Forman and Singpurwalla (1977) in the sense that we have a numerical measure to decide to stop or not debugging the software.

To illustrate the performance of our proposed procedure to stop testing the software, we consider the data obtained during the debugging of a data reduction program called the F 11-D program. This program consists of "approximately 3-4 thousand" FORTRAN statements. As can be seen from Table 4.2, a total of 107 errors were detected after debugging the program for a total of 226.11 seconds of CPU time. Several strategies for distributing the errors in each time interval are available to a user. In our case, we adopt the strategy in which the times between errors in each interval is the quotient of the interval length (CPU time) to the number of errors in the interval. For more details about this data see Moranda (1975) or Forman and Singpurwalla (1977)

In Table 4.3, the successive values of $d(n,10)$ and the posterior mean are computed at the end of each interval to decide to stop debugging the software. We choose $c_0 = 10$ to be sure we have a quite asymmetric loss

function, however, we can verify that the optimal solution under the LINEX function is very stable for $c_0 \geq 1$.

At the end of each interval, we compute $d(n,10)$ to see the influence of the LINEX function on the estimative of N. For example, at the end of the interval number five we have a strong influence of the LINEX function with respect to posterior mean and our measure $d(40,10)$ signals us to continue debugging the software.

Table 4.2: Data on F 11-D Program

Interval number	Date	Number of errors: n	Cumulative number of errors	CPU time (seconds)
1	1/12	8	8	0.5
2	1/15	7	15	0.6
3	1/16	1	16	0.65
4	1/17	8	24	1.9
5	1/18	16	14	1.59
6	1/19	18	58	8.83
7	1/22	13	71	9.94
8	1/23	8	79	7.25
9	1/24	9	88	8.34
10	1/25	2	90	3.86
11	1/26	6	96	13.11
12	1/27	3	99	34.15
13	1/29	3	102	82.7
14	1/30	2	104	1.1
15	1/31	3	107	51.59
Total		107		226.11

After the twelfth interval, our procedure shows a better agreement than we had before and gives us a good indication that the debugging process is close to finish. In addition to obtain robustness with respect to the LINEX function after having observed $n = 99$, Forman and Singpurwalla (1977) showed for this data a very good agreement between the normal and the actual relative likelihood functions.

4.4 CONCLUSIONS

An important aspect of our proposed stopping procedure is the robustness with respect to the shape parameter c of the LINEX function. It means that our Bayesian procedure to estimate N is not affected when replacing the squared loss function by the LINEX loss function.

Table 4.3 Stopping rule for debugging the software using F11-D data

n	$d(n,10)$	$E(N \mid t^{(n)})$	$\hat{N}_{MLE}$
8	99.64	124.30	9
15	108.25	140.81	16
16	31.51	53.19	17
24	32.59	61.55	24
40	163.13	226.42	43
58	12.04	70.74	60
71	7.72	79.18	73
79	7.51	86.98	81
88	8.69	97.24	90
90	7.08	97.55	92
96	4.46	99	99
99	0.66	99	100
102	0.03	102	102
104	0.08	104	104
107	0.05	107	107

Also, as discussed by Forman and Singpurwalla (1977), for small values of $d(n,10)$ we have the normality of the relative likelihood function. Also, the Bayes procedure provides a very stable estimative of N compared with the MLE.

CHAPTER 5

PARAMETER ESTIMATION OF SOME NHPP SOFTWARE RELIABILITY MODELS WITH CHANGE-POINT

5.1 INTRODUCTION

As the software systems play an increasingly important role in computer systems, intensive studies have been carried out to ensure the software reliability. During the past three decades many software reliability growth models (SRGMs) have been proposed; see Musa et.al. (1987), Xie (1991), Pham (1999), Singpurwalla and Wilson (1999), among many others. A very important class is Non Homogeneous Poisson Process (NHPP) models, which has great advantages in practice and attracted a lot of attention. The main issue in the NHPP model is to determine an appropriate failure intensity function. Once the failure intensity function is determined, the software reliability and the related measurements can be easily derived.

Various NHPP software reliability models have been built upon various assumptions for the testing process. For many NHPP models it is usually assumed that each time a failure occurs, the fault that cause the failure can be immediately removed and no new faults are introduced, which is usually called perfect debugging, and that during the testing process the fault detection rate is a constant in the first NHPP model (GO model see Goel and Okumoto, 1979). However, in many realistic situations, the software-testing process can be affected by many factors, such as the running environment, testing strategy and resource allocation. Once these factors are changed during the testing phase, the software failure intensity may increase or decrease nonmonotonically. It is a change point problem proposed by Zhao (1993) in software reliability estimation the change

point effect should be considered simultaneously. Otherwise, the estimated model may not express the factual software reliability behavior.

Recently, the change-point problem in software failure process is considered by Zou (2003) and some NHPP software reliability models with change-point have been proposed. The change-point of the fault detection rate in the NHPP model is first considered in Chang (2001). Moreover, Shyur (2003) incorporated both imperfect debugging and change-point problem into the NHPP model. The unknown change-point in the model should be estimated from the observed data. In statistical literature, although the change-point problem has been studied by many authors (see Nguyen et.al., 1984; Yao, 1986; Pham and Nguyen, 1990; Muller, 1992; Loader, 1996; Chen and Gupta,2001; Fotopoulos and Jandhyala, 2001; etc.), the NHPP software reliability model with change-point is not considered much. To find the intensity functions with change-point, Chang (2001) used the least square method to estimate the parameter assuming that the change-point is located at the failure data points and that the functional form of the intensity function of the NHPP is known. However, in many realistic situations the change is not necessarily occurring at some failure points, we use the maximum likelihood method to estimate the change-point and the other model parameters. If there is not much information on the testing process or even the failure intensity function is completely unknown, the maximum likelihood method is not applicable. Then we propose a nonparametric method to estimate the change-point of NHPP. This nonparametric estimation can reduce the modeling bias as well. The simulation results show that the methods work well.

5.2 NHPP MODELS WITH CHANGE-POINT

Many NHPP models are very useful to describe the software failure process which possesses certain trends such as reliability growth. In this chapter we review the GO model proposed by Goel and Okumoto (1979) and the imperfect-software-debugging model proposed by Pham (1993). The extensions of these two models with change-point are also reviewed.

5.2.1 GO MODEL

Software testing processes are often modeled as a fault counting process. Let $\{N(t), t \geq 0\}$ be a counting process representing the cumulative number of software failures by time t. The process $N(t)$ is shown to be a NHPP with mean value function $m(t)$ and failure intensity function $\lambda(t)$. Goel and Okumoto (1979) assume that the software failure intensity $\lambda(t)$ is proportional to the expected number of undetected failures. Thus, $m(t)$ can be obtained by solving the following differential equation:

$$\lambda(t) = \frac{dm(t)}{dt} = b[a - m(t)] \tag{5.2.1}$$

where a denotes the initial number of faults contained in the software and b is known as the fault detection rate. Solving the equation, the mean value function and the intensity function are obtained as follows:

$$m(t) = a(1 - e^{-bt}), \tag{5.2.2}$$

and

$$\lambda(t) = abe^{-bt} \tag{5.2.3}$$

In software reliability, the initial number of faults and the fault detection rate are always unknown. The parameters can be evaluated by some statistical methods.

5.2.2 IMPERFECT-SOFTWARE –DEBUGGING MODEL

Following the GO model, the constant a implies the perfect debugging assumption, i.e., no new faults are introduced during the debugging process. Pham (1993) introduced a NHPP software reliability model that is subject to imperfect debugging. This model is an extension of the GO model. He assumed if detected faults are removed, then there is a possibility to introduced new faults with a constant rate β. Let $\alpha(t)$ be the number of faults to be eventually detected (denoted by a) plus the number of new faults introduced to the software by time t, the mean value function $m(t)$ can be given as the solution of the following system of differential equations:

$$\lambda(t) = \frac{\partial m(t)}{\partial t} = b[\alpha(t) - m(t)],$$

$$\frac{\partial \alpha(t)}{\partial t} = \beta \frac{\partial m(t)}{\partial t},$$

$$\alpha(0) = a, m(0) = 0 \qquad (5.2.4)$$

where a is the number of faults to be eventually detected. Solving the above equations, we can obtain the mean value function and intensity function, respectively, as follows

$$m(t) = \frac{a}{1-\beta}[1 - e^{-(1-\beta)bt}], \qquad (5.2.5)$$

and

$$\lambda(t) = abe^{-(1-\beta)bt}. \qquad (5.2.6)$$

5.2.3 GO MODEL WITH CHANGE-POINT

It is quite often that the fault detection rate b is supposed to be a constant in many NHPP software reliability models. And the failure intensity function $\lambda(t)$ is often expected to be a constant function of time. However, this fault detection rate may be affected by many factors, such as the testing strategy, resource allocation and so on. During a software testing process, there is possibility that the underlying fault detection rate function is changing at some moment τ, called "change-point". Considering the change-point in software models is intended to be close to the reality.

Chang (2001) considered the GO model with change-point, which is an extension of the GO model. Then the fault detection rate b is not a constant now, and it is supposed to have a change-point. Therefore, the fault detection rate at testing time t can be defined as

$$b(t)=\begin{cases} b_1 & 0 \le t \le \tau \\ b_2 & t > \tau \end{cases} \tag{5.2.7}$$

Where τ is the change-point, a is the expected number of faults contained in the software at the beginning of test, and b_1, b_2 are the fault detection rate before and after the change-point respectively.

Under these assumptions, solving the equation:

$$\lambda(t)=\frac{dm(t)}{dt}=b(t)[a-m(t)], \tag{5.2.8}$$

The mean value function and intensity function can be derived as

$$m(t)=\begin{cases} a(1-e^{-b_1 t}) & 0 \le t \le \tau, \\ a[1-e^{-b_1\tau-b_2(t-\tau)}] & t > \tau, \end{cases} \tag{5.2.9}$$

and

$$\lambda(t)=\begin{cases} ab_1e^{-b_1t} & 0\le t\le \tau, \\ ab_2e^{-b_1\tau-b_2(t-\tau)} & t>\tau, \end{cases} \tag{5.2.10}$$

It should be noticed that $m(t)$ is a continuous function, but $\lambda(t)$ is discontinuous at $t=\tau$

5.2.4 IMPERFECT-SOFTWARE-DEBUGGING MODEL WITH CHANGE-POINT

The imperfect-software-debugging model supposes the fault detection rate b and the fault introduction rate β are constants. However, in practice, there is a possibility that these rates are changing at some moment. Shyur (2003) extended the imperfect-software-debugging model with change-point problem. The fault detection rate and fault introduction rate are supposed to have a change-point in software testing phase. Therefore, the fault detection rate function at testing time t is

$$b(t)=\begin{cases} b_1 & 0\le t\le \tau \\ b_2 & t>\tau \end{cases} \tag{5.2.11}$$

Where τ is the change-point and the fault introduction rate $\beta(t)$ can be defined as

$$\beta(t)=\begin{cases} \beta_1 & 0\le t\le \tau \\ \beta_2 & t>\tau \end{cases} \tag{5.2.12}$$

Solving the equations

$$\lambda(t)=\frac{dm(t)}{dt}=b(t)[\alpha(t)-m(t)],$$

$$\frac{\partial\alpha(t)}{\partial t}=\beta(t)\frac{\partial m(t)}{\partial t}, \qquad \alpha(0)=a,\ m(0)=0 \tag{5.2.13}$$

the mean value function and intensity function can be derived as

$$m(t)=\begin{cases}\dfrac{\alpha}{1-\beta_1}[1-e^{-(1-\beta_1)b_1 t}] & 0\le t\le \tau \\ \dfrac{\alpha}{1-\beta_2}[1-e^{-(1-\beta_1)b_1\tau-(1-\beta_2)b_2(t-\tau)}]+\dfrac{m(\tau)(\beta_1-\beta_2)}{1-\beta_2} & t>\tau\end{cases} \tag{5.2.14}$$

and

$$\lambda(t)=\begin{cases}ab_1 e^{-(1-\beta_1)b_1 t} & 0\le t\le \tau \\ ab_2 e^{-(1-\beta_1)b_1\tau-(1-\beta_2)b_2(t-\tau)} & t>\tau\end{cases} \tag{5.2.15}$$

5.3 MAXIMUM LIKELIHOOD ESTIMATION

The software reliability models involve unknown parameters, which are to be estimated from the observed data. In the NHPP models with change-point, the change-point in testing process is always unknown. In this chapter we propose a maximum likelihood method to estimate the change-point and other parameters of the NHPP models under the condition that the functional form of the intensity function is known. Here we only consider the GO model with change-point. The estimation procedures for other NHPP models with change-point are similar.

5.3.1 FAILURE TIME DATA

Suppose the software testing interval lasted T units of time and n faults were observed. Let $0<S_1<S_2<\cdots<S_n\le T$ be the time at which the failures were observed. From the NHPP properties, the likelihood function of the observed data is

$$L(\tau,a,b_1,b_2)=\exp\{-a[1-e^{-b_1\tau-b_2(T-\tau)}]\}\prod_{i=1}^{N(\tau)}ab_1e^{-b_1S_i}$$
$$\prod_{i=N(\tau)+1}^{n}ab_2e^{-b_1\tau-b_2(S_i-\tau)} \tag{5.3.1}$$

Thus the log likelihood function is

$$\begin{aligned}L^* &= \log L(\tau,a,b_1,b_2)\\ &= -a[1-e^{b_1\tau-b_2(T-\tau)}]\\ &\quad+\sum_{i=1}^{N(\tau)}(\log ab_1-b_1S_i)+\sum_{i=N(\tau)+1}^{n}[\log ab_2-b_1\tau-b_2(S_i-\tau)]\end{aligned} \tag{5.3.2}$$

By Nguyen et.al. (1984), the log likelihood function tends to infinity as the change-point τ tends to the failure time S_n from below. Since the log likelihood function is unbounded, the estimate of τ can not be obtained by maximizing the log likelihood function over $\tau\in[0,T]$. In practice, the change-point is supposed to be in the testing phase but not near the boundary. Therefore, it is reasonable to assume that there are failures occurring before and after the change. This leads us to restrict the change-point in the interval $[S_2,S_{n-1}]$. Hence, log likelihood function is bounded when $\tau\in[S_2,S_{n-1}]$ and the estimate of τ can be obtained by maximizing the log likelihood function. The estimates of the parameters $\hat{\tau},\hat{a},\hat{b}_1,\hat{b}_2$ should satisfy

$$L^*(\hat{\tau},\hat{a},\hat{b}_1,\hat{b}_2)=\max_{\tau\in[S_2,S_{n-1}]}\max_{a,b_1,b_2}L^*(\tau,a,b_1,b_2) \tag{5.3.3}$$

If $\tau\in[S_k,S_{k+1})$, then $N(\tau)=k$,so the log likelihood function is

$$L_k^*(\tau,a,b_1,b_2)=-a[1-e^{-b_1\tau-b_2(T-\tau)}]+\sum_{i=1}^{k}(\log ab_1-b_1Si)$$
$$+\sum_{i=k+1}^{n}[\log ab_2-b_1\tau-b_2(S_i-\tau)], \tag{5.3.4}$$

where $a>0, b_1>0, b_2>0$. Since L_k^* is a continuous function, L_k^* admits a maximum on $[S_k, S_{k+1}]$ either at some point in $[S_k, S_{k+1}]$ or at the failure point S_{k+1}. The maximum value can be obtained by numerical computation. Denote the optimal solution by τ_k, a_k, b_{1k}, b_{2k}, respectively, and the maximum value over $[S_k, S_{k+1}]$ is

$$L_k^*(\tau_k, a_k, b_{1k}, b_{2k}) = \max_{\tau \in [S_k, S_{k+1}]} \max_{a, b_1, b_2} L_k^*(\tau, a, b_1, b_2) \tag{5.3.5}$$

Therefore, the maximum value over the whole interval $[S_2, S_{n-1}]$ is

$$\begin{aligned} L^*(\hat{\tau}, \hat{a}, \hat{b}_1, \hat{b}_2) &= \max_k \max_{\tau \in [S_k, S_{k+1}]} \max_{a, b_1, b_2} L_k^*(\tau, a, b_1, b_2) \\ &= \max_k L_k^*(\tau_k, a_k, b_{1k}, b_{2k}) \end{aligned} \tag{5.3.6}$$

Then the estimates $\hat{\tau}$, $\hat{a}$, $\hat{b}_1$, $\hat{b}_2$, can be obtained by comparing the values of $L_k^*(\tau_k, a_k, b_{1k}, b_{2k})$.

5.3.2 GROUPED DATA

Sometimes the failures can not be observed exactly and only the number of failures up to a given time is known. This type of data is known as grouped data, which is widely studied in software reliability engineering. In this case the likelihood function is not in the form of (5.3.1). thus, a new form of the maximum likelihood estimation is required. This is going to be given below.

Suppose we observed the software failure process $\{N(t)\}$ at n time points: $0 = S_0 < S_1 < S_2 < \cdots < S_n = T$. Let y_i be the total number of the observed failures up to the time S_i, $i = 0,1,\ldots,n$. So $y_i - y_{i-1}$ is the number of observed failures occurring during the interval $[S_{i-1}, S_i]$. The likelihood function of the observed data is

$$L(\tau,a,b_1,b_2)=\prod_{i=1}^{n}\frac{[m(S_i)-m(S_{i-1})]^{y_i-y_{i-1}}}{(y_i-y_{i-1})!}\exp\{-m(T)\}, \quad (5.3.7)$$

And the log likelihood function is

$$\begin{aligned} L^*(\tau,a,b_1,b_2)=&\sum_{i=1}^{n}(y_i-y_{i-1})\log[m(s_i)-m(s_{i-1})] \\ &-\sum_{i=1}^{n}\log[(y_i-y_{i-1})!]-m(T) \end{aligned} \quad (5.3.8)$$

The change-point is also supposed to bee not near the boundary and we restrict it in the interval $[S_2,S_{n-1}]$. Then the estimates $\hat{\tau}$, $\hat{a}$, $\hat{b}_1$, $\hat{b}_2$ should satisfy

$$L^*(\hat{\tau},\hat{a},\hat{b}_1,\hat{b}_2)=\max_{\tau\in[S_2,S_{n-1}]}\max_{a,b_1,b_2}L_k^*(\tau,a,b_1,b_2) \quad (5.3.9)$$

If $\tau\in[s_k,S_{k+1}]$, the log likelihood function is

$$\begin{aligned} L^*(\tau,a,b_1,b_2)=&-a[1-e^{-b_1\tau-b_2(T-\tau)}]-\sum_{i=1}^{n}\log(y_i-y_{i-1})! \\ &+(y_{k+1}-y_k)\log[ae^{-b_1S_k}-ae^{-b_1\tau-b_2(S_{k+1}-\tau)}] \\ &+\sum_{i=k+2}^{n}(y_i-y_{i-1})\log[ae^{-b_1\tau-b_2(S_{i-1}-\tau)}-ae^{-b_1\tau-b_2(S_i-\tau)}] \\ &+\sum_{i=1}^{k}(y_i-y_{i-1})\log[ae^{-b_1S_{i-1}}-ae^{-b_1S_i}] \end{aligned} \quad (5.3.10)$$

Since L_k^* is a continuous function, L_k^* admits a maximum on $[S_k,S_{k+1}]$either at some point in $[S_k,S_{k+1}]$ or at the failure point S_{k+1}.

The optimal solution $\tau_k, a_k, b_{1k}, b_{2k}$ can be obtained by numerical computation and the maximum value over $[S_k, S_{k+1}]$ is

$$L_k^*(\tau_k, a_k, b_{1k}, b_{2k}) = \max_{\tau \in [S_k, S_{k+1}]} \max_{a, b_1, b_2} L_k^*(\tau, a, b_1, b_2) \tag{5.3.11}$$

So the maximum value over the whole interval $[S_2, S_{n-1}]$ is

$$\begin{aligned} L^*(\hat{\tau}, \hat{a}, \hat{b}_1, \hat{b}_2) &= \max_k \max_{\tau \in [S_k, S_{k+1}]} \max_{a, b_1, b_2} L_k^*(\tau, a, b_1, b_2) \\ &= \max_k L_k^*(\tau_k, a_k, b_{1k}, b_{2k}) \end{aligned} \tag{5.3.12}$$

Then the estimates $\hat{\tau}$, $\hat{a}$, $\hat{b}_1$, $\hat{b}_2$ can be obtained by comparing the values of $L_k^*(\tau_k, a_k, b_{1k}, b_{2k})$.

5.4 NONPARAMETRIC ESTINMATION

In the previous section, change-point estimation is based on the assumption that the functional form of the intensity function is known. However, this is very often not the case. Then the choice of a model will sometime cause a large bias. In this situation we propose a non-parametric method to estimate the change-point.

5.4.1 FAILURE TIME DATA

Suppose the failure intensity function $\lambda(t)$ is continuous during the testing phase $[0, T]$ except at a change-point τ. That is, the intensity function is discontinuous only at τ. Suppose n failures are observed in $[0, T]$, and let $0 < S_1 < S_2 < \cdots < S_n \le T$ be the failure times.

If the failure intensity function is continuous in the whole interval, a non-parametric estimation for $\lambda(t)$ is

$$\hat{\lambda}(t)=\frac{1}{h}\sum_{i=1}^{n}K\left(\frac{S_i-t}{h}\right), \tag{5.4.1}$$

where $K(u)$ is a kernel function and h is the bandwidth.

To estimate the change-point, we investigate the maximum difference between left and right estimates, where the right estimate $\hat{\lambda}_+(t)$ is estimated using observations located at the right of point t, and the left estimate $\hat{\lambda}_-(t)$ is estimated using the data located at the left. Accordingly, an estimate for a possible jump $\hat{\Delta}(t) = \hat{\lambda}_+(t) - \hat{\lambda}_-(t)$ follows. Then the estimate $\hat{\tau}$ is defined as the point where the possible jump achieves maximum.

Suppose the change-point is not at the boundary and there are observed failures before and after change. We can restrict the change-point in the interval $[S_l, S_{n+1-l}]$, here l is some integer. Suppose $K_+(.)$ is a positive kernel function with support [0,1] and $K_-(.)$ is defined by $K_-(u)=K_+(-u)$. A basic assumption is that $K_+(0)>0, K(u)\geq 0$ when $0<u\leq 1$, $\int_0^1 K(u)du=1$. then the left and right estimates of $\lambda(t)$ are

$$\hat{\lambda}_-(t)=\frac{1}{h_-}\sum_{i=1}^{n}K_-\left(\frac{S_i-t}{h_-}\right)$$

and

$$\hat{\lambda}_+(t)=\frac{1}{h_+}\sum_{i=1}^{n}K_+\left(\frac{S_i-t}{h_+}\right),$$

respectively, where h_- and h_+ are left and right bandwidths. Then the estimate $\hat{\tau}$ should satisfy

$$\hat{\Delta}(\hat{\tau})=\max_{t\in[S_l,S_{n+1-l}]}\hat{\Delta}(t)=\max_{t\in[S_l,S_{n+1-l}]}[\hat{\lambda}_+(t)-\hat{\lambda}_-(t)] \tag{5.4.2}$$

if $t \in [S_k, S_{k+1}]$, we can choose bandwidth $h_- = S_{k+1} - S_{k-j}$ and $h_+ = S_{k+1+j} - S_k$, here j is an integer number. Thus $\hat{\Delta}(t)$ on $[S_k, S_{k+1}]$ is

$$\hat{\Delta}_k(t) = \frac{1}{h_+} \sum_{i=k+1}^{n} K_+\left(\frac{S_i - t}{h_+}\right) - \frac{1}{h_-} \sum_{i=1}^{k} K_-\left(\frac{S_i - t}{h_-}\right). \qquad (5.4.3)$$

Since $\hat{\Delta}_k(t)$ is a continuous function in the interval $[S_k, S_{k+1}]$, let t_k be the point where $\hat{\Delta}_k(t)$ admits a maximum on $[S_k, S_{k+1}]$. Then the maximum value of $\hat{\Delta}_k(t)$ on $[S_k, S_{k+1}]$ is

$$\hat{\Delta}_k(t) = \max_{t \in [S_k, S_{k+1}]} \hat{\Delta}_k(t). \qquad (5.4.4)$$

Therefore, the maximum value of $\hat{\Delta}(t)$ on the whole interval $[S_l, S_{n+1-l}]$ is

$$\hat{\Delta}(\hat{\tau}) = \max_k \max_{t \in [S_k, S_{k+1}]} \hat{\Delta}_k(t) = \max_k \hat{\Delta}_k(t_k). \qquad (5.4.5)$$

Then the estimate $\hat{\tau}$ is obtained by comparing the values of $\hat{\Delta}_k(t_k)$.

5.4.2 GROUPED DATA

When the failure data is grouped, we propose the following non-parametric method to estimate the change-point. Suppose we observe the software failure process $\{N(t)\}$ at n time points, $0 = S_0 < S_1 < S_2 < \cdots < S_n = T$. Let y_i be the total number of observed failures up to the time S_i, $i = 0,1,..,n$.so $y_i - y_{i-1}$ is the number of observed failures occurring during the interval $(S_{i-1}, S_i]$. We also consider the maximum difference between the left and right estimates. If $t \in (S_k, S_{k+1}]$, the left and right estimates of $\lambda(t)$ are

$$\hat{\lambda}_k^-(t) = \sum_{i=1}^{k+1} W_i^-(t)\left(\frac{y_i - y_{i-1}}{S_i - S_{i-1}}\right) / \sum_{i=1}^{k+1} W_i^-(t)$$

and

$$\hat{\lambda}_k^+(t) = \sum_{i=k+1}^{n} W_i^+(t)\left(\frac{y_i - y_{i-1}}{S_i - S_{i-1}}\right) / \sum_{i=k+1}^{n} W_i^+(t),$$

respectively, where

$$W_i^-(t) = \int_{S_{i-1}}^{S_i} K_-\left(\frac{u-t}{h_-}\right) du$$

and

$$W_i^+(t) = \int_{S_{i-1}}^{S_i} K_+\left(\frac{u-t}{h_+}\right) du$$

are left and right interval weights. Therefore, $\hat{\Delta}(t)$ on the interval $t \in (S_k, S_{k+1}]$ is

$$\hat{\Delta}_k(t) = \hat{\lambda}_k^+(t) - \hat{\lambda}_k^-(t)$$

$$= \sum_{i=k+1}^{n} \int_{S_{i-1}}^{S_i} K_+\left(\frac{u-t}{h_+}\right) du \left(\frac{y_i - y_{i-1}}{S_i - S_{i-1}}\right) / \sum_{i=k+1}^{n} \int_{S_{i-1}}^{S_i} K_+\left(\frac{u-t}{h_+}\right) du$$

$$- \sum_{i=1}^{k+1} \int_{S_{i-1}}^{S_i} K_-\left(\frac{u-t}{h_-}\right) du \left(\frac{y_i - y_{i-1}}{S_i - S_{i-1}}\right) / \sum_{i=1}^{k+1} \int_{S_{i-1}}^{S_i} K_-\left(\frac{u-t}{h_-}\right) du. \quad (5.4.6)$$

The function $\hat{\Delta}_k(t)$ is a continuous function in the interval $[S_k, S_{k+1}]$. Let t_k be the point where $\hat{\Delta}_k(t)$ admits a maximum on $[S_k, S_{k+1}]$. Then the maximum value of $\hat{\Delta}_k(t)$ on $[S_k, S_{k+1}]$ is

$$\hat{\Delta}_k(t_k) = \max_{t \in [S_k, S_{k+1}]} \hat{\Delta}_k(t) \quad (5.4.7)$$

Table 5.1: Software failure times data: System T 1

Software failure times							
3	1846	5324	10258	15806	26770	42296	56485
33	1872	5389	10491	16185	17753	42296	56560
146	1986	5565	10625	16229	28460	45406	57042
227	2311	5623	10982	16358	28493	46653	62551
342	2366	6080	11175	17168	29361	47596	62651
351	2608	6380	11411	17458	30085	48296	62661
353	2676	6477	11442	17758	32408	49171	63732
444	3098	6740	11811	18287	35338	49416	64106
556	3278	7192	12559	18568	36799	50145	64893
571	3288	7447	12559	18728	37642	52042	71043
709	4434	7644	12791	19556	37654	52489	74364
759	5034	7837	13121	20567	37915	52875	75409
836	5049	7843	13486	21012	39715	53321	76057
860	5085	7922	14708	21308	40580	53443	81542
968	5089	8738	15251	23063	42015	54433	82702
1056	5089	10089	15261	24127	42045	55381	84566
1726	5097	10237	15277	25910	42188	56463	88682

Therefore, the maximum value of $\hat{\Delta}(t)$ on the whole interval $[S_{\mathfrak{l}}, S_{n+1-l}]$ **is**

$$\hat{\Delta}(\hat{\tau}) = \max_{k} \max_{t\in[S_k, S_{k+1}]} \hat{\Delta}_k(t) = \max_{k} \hat{\Delta}_k(t_k) \qquad (5.4.8)$$

Then the non-parametric estimate $\hat{\tau}$ can be obtained by comparing the values of $\hat{\Delta}_k(t_k)$.

5.5 EXAMPLE

Table 5.1 shows the data set collected from the system T 1 in Musa et al. (1987). This data set includes 136 faults found in the testing phase. We use the GO model and GO with change-point model to fit the failure data, respectively. The GO with change-point model fit of this data resulted in parameter estimates of $\hat{\tau}=1056$, $\hat{a}=144.4$, $\hat{b}_1=1.11\times10^{-4}$, and $\hat{b}_2=2.99\times10^{-5}$. It is clear that the estimates $\hat{b}_1$ and $\hat{b}_2$ are significantly different, and there is a change in the testing process. We separate the data into two subsets.

Table 5.2: Comparison of descriptive and predictive power of GO models

	With change-point	No change-point
Maximize log-likelihood	-828.8	-836.9
SSE(fit)	2939.895	6858.424
SSE(predict)	21.778	162.327

Table 5.3: Mean absolute error for ML estimators for failure time data case

a	$\hat{\tau}$	$\hat{a}$	$\hat{b}_1$	$\hat{b}_2$
1000	0.00190	25.7152	0.0429	0.2119
500	0.0022	19.6859	0.0661	0.3051
250	0.0051	14.1699	0.0922	0.4390

The first 120 records are used to fit the models and estimate the parameters; the remaining points are used to illustrate the predictive power of the models. Table 5.2 summarizes the SSE values for the models. It shows that the SSE-fit and SSE-predict values for the change-point model are smaller than other ones which do not consider the change-point problem. Therefore, we suggest that there is a change in the testing process after 1056 CPU time of the testing phase according to the observed failures.

5.6 SIMULATION STUDY

To evaluate the performance of the proposed estimation methods, some simulation data are studied. It was designed to test using a length of time $T=1$, the expected number of software faults at the beginning of test $a=250,500$ or 1000, and the change-point τ is 0.4, for incomplete failure data, we choose $k=50$ and $S_i=i/k$, $i=1,2,\ldots,k$. Here, we take $b_1=1$ and $b_2=4$. Table 5.3 and 5.4 give the estimators by maximum likelihood methods for the complete failure data and incomplete data, respectively. In each table we repeated 1000 times for each case.

Table 5.4: Mean absolute error for ML estimators for grouped data case

a	$\hat{\tau}$	$\hat{a}$	$\hat{b}_1$	$\hat{b}_2$
1000	0.0018	28.1146	0.0422	0.2262
500	0.0027	19.5860	0.0677	0.3114
250	0.0052	14.0985	0.0978	0.4596

Table 5.5: Mean absolute error for nonparametric estimators for failure time data case

a	Mean absolute error
1000	0.0061
500	0.0084
250	0.0142

The comparison criteria are mean absolute error. It is clear that the maximum likelihood estimators have small mean absolute error for both cases of complete data and incomplete data. Therefore, the maximum likelihood estimators behave satisfactorily in all cases.

As an illustration of the non-parametric methods, we consider the failure data, following the same model as above. The kernel function we use is $K_+(u) = \frac{3}{2}(1-u^2), 0 \le u \le 1$ and $K_-(u) = K_+(-u), -1 \le u \le 0$, for complete failure data, we choose the bandwidth $h_- = S_{k+1} - S_{k-40}$ and $h_+ = S_{k+41} - S_k$. And for incomplete data, we take $h_- = S_{k+1} - S_{k-10}$ and $h_+ = S_{k+11} - S_k$. Table 5.5 gives the non-parametric estimator for the complete data, while table 5.6 gives the non-parametric estimator for the incomplete data. In each table we repeated 1000 times for each case. The comparison criteria are mean absolute error. It is shown that the non-parametric estimators have small mean absolute error for both cases.

Therefore, the non-parametric estimators behave satisfactorily when not much information is known.

5.7 DISCUSSION

The maximum likelihood method and non-parametric method for estimating the change-point of a NHPP model have been proposed. Once the model parameters determined the software reliability easily can be derived. The result of simulation in section 5.5 shows that the estimators perform well. This maximum likelihood method is useful for various NHPP with change-point software reliability models. If the failure intensity function is completely unknown, the non-parametric method for estimating the change-point behaves satisfactorily.

Recently the change-point problems in statistical models are attracting a lot of attention. Statistical inference for change-point is becoming more and more important. Various types of change-point problems in various models are widely considered; for example, see Aue and Steinebach (2002), Pons (2002), Dabye et al. (2003), and Xie et al. (2004). In this chapter we only consider that there is one change-point in the testing phase. In a realistic situation, it is possible that there is more than one change-point. Therefore, the NHPP model with two or more change-points problem should be further studied.

In the GO model with change-point, the fault detection rate is changing at some moment abruptly. However, sometimes, the parameter is changing gradually in some interval. For example, the fault detection rate is

$$b(t)=\begin{cases} b_1 & 0 \le t \le \tau_1, \\ b_1 + \dfrac{b_2 - b_1}{\tau_2 - \tau_1}(t-\tau_1) & \tau_1 < t \le \tau_2, \\ b_2 & t > \tau_2, \end{cases}$$

where τ_1 and τ_2 are change-points and b_1, b_2 are constants. The NHPP models with this type of change-point should be considered and statistical methods should be further developed.

Table 5.6: Mean absolute error for nonparametric estimators for grouped data case

a	Mean absolute error
1000	0.0061
500	0.0084
250	0.0142

Furthermore, whether a change-point exists is always unknown, so it is of interest to test $H_0 : b_1 = b_2$ versus $H_1 : b_1 \neq b_2$ in the GO model with change-point. Moreover, the test for other parameters with a change-point in the NHPP models is also important in practice. Therefore, the testing procedure in the NHPP models with change-point should be developed.

REFERENCES

CHAPTER-1

Blake, I. F., (1979). *An Introduction to Applied Probability.* Wiley, New York.

Chang, D. S, Huang, D.Y. and Tseng, S. T. (1992). Selecting the Most Reliable Design under Type-II Censored Accelerated Tests. *IEEE Transactions on Reliability, 41:588-592.*

Lawless, J. F. (1982). *Statistical Models and Methods for Lifetime* Data. John Wiley, New York.

Mann, N. R. Schafer, R. E. and Singpurwalla, N. D. (1974). *Methods for Statistical Analysis of reliability and Life Data.* John Wiley, New York.

Martz, H. F. Waller, R. A. (1982). *Bayesian Reliability Analysis.* John Wiley, New York.

Sinha, S. K. (1985). *Reliability and Life Testing.* Wiley-Eastern. www.Weibull.com

Nelson, W. (1990). *Accelerated Testing, Statistical Models, Test Plans and Data Analysis.* Wiley, New York.

Nelson, W. and Meeker, W. Q. (1978). Theory for Optimum Accelerated Censored Life Tests for Weibull and Extreme Value Distributions *Technometrics,* **20**:171-177.

Sinha, S. K. (1985). *Reliability and Life Testing.* Wiley-Eastern.

Tseng, S. T. and Chang, D. S. (1989). Selecting the Most Reliable Design under Type-II Censoring. *Reliability Engineering and System Safety* ***25****:147-156.*

Tseng, S. T., Huang, D. Y. and WU, T. Y. (1994). A Sampling Plan for Selecting The Most Reliable Product Under The Arrhenius Accelerated Life Tests Model. *Statistical Sinica* **41**:215-230.

Tseng, S. T. (1994). Planning Accelerated Life Tests for Selecting the Most Reliable Product. *J. Statist. Plann.and Inference* **41**:215-230.

CHAPTER 2

DTU Course catalogue 2002(Technical University of Denmark) Course-02445).

Malaiya, Y.K. and P. Srimani: "Software Reliability Models", *IEEE Computer Society Press, 1990*

Mathur A.P., Vernon J. Regoand Krishnamurthy S., "White-Box Models for the Estimation of Software Reliabiliy".

Musa J.D. Iannino A. and Okumoto K. (1987), "*Software Reliability: Measurement , Prediction, and Application*", McGraw-Hill, New York.

CHAPTER 3

Caristein, E. (1988). Nonparametric change-point estimation. *Ann. Statist.* **16**, 188-197.

Dawid, A.P. (1984). Statistical theory: the prequential approach. *J. Royal Statist. Soc.* **A**, 147, 278-292

Hinkley, D.V. (1970). Inference about the change-point in a sequence of random variables. *Biometrika*, **57**, 1-16.

Hinkley, D. V. and Hinkley, E. A. (1970). Inference about the change-point in a sequence of binomial variables. *Biometrika*, **57**, 477-488.

Huang, X. Z. (1990). The limit conditions of some time between failure models of software reliability. *Microelectronics and Reliability*, **30**, 481-485.

Jelinski, Z. and Moranda, P.B. (1972). *Software reliability research, in Statistical Computer Performance Evaluation, Ed. Freiberger, W.*, Academic Pres, New York, 465-497.

Joe, H. and Reid, N. (1985). Estimating the number of faults in a system. *J. American Statistical Association,* **80**, 222-226.

Joseph, L. and Wolfson, D. B. (1992). Estimation in multi-path change-point problems. *Commun. Statist. Theory and Method.*, 21, 897-913.

Littlewood, B. (1981). Stochastic reliability growth: model for fault-removal in computer programs and hardware-design. *IEEE Trans. Reliability, R-30, 312-320.*

Littlewood, B. (1991). Modeling growth in software reliability. In Software Reliability Handbook. Ed Paul Rook. *Elsevier Applied Science.* 111-136.

Musa, J. D.; Iannino, A. and Okumoto, K. (1987). *Software Reliability: Measurement, Prediction, Application.* McGraw-Hill, New York.

Musa, J. D. (1979). *Software Reliability Data.* RADC, New York.

Schick, G. J. and Wolverton, R. W. (1973). *Assessment of software reliability. Proc. Operations Research*, Physica-Verlag, Wurzburg wein, 395-422.

Wright, D. E. and Hazelhurst, C. E. (1988). Estmation and prediction for a simple software reliability model. *Statistician*, **37**, 319-325.

CHAPTER 4

Forman, E. H. and Singpurwalla, N. D. (1977). "an empirical stopping rule for debugging and testing computer software," J. American Statistical Assoc., **72**, 360, 750-757.

Goel, A. L. and Okumoto, K. (1979). "Time-dependent error-detection rate model for software reliability and other performance measures." *IEEE trans. Reliability, R-28,206-211.*

Jelinski, Z. and Moranda, P.B. (1972). "*Software reliability research. In Statistical Computer Performance Evaluation.*" New York and London: Academic Press, 465-484.

Littlewood, B. (1981). "A critique of the Jelinski-Moranda model for software reliability," *Proc. Ann. Rel. and Maintainability Symp.*, 357-364.

Littlewood, B and Verral, J. L. (1973). "A Bayesian reliability growth model for computer software," *Record IEEE Symposium on computer Software Reliability, 70-77.*

Langberger, N. and Singpurwalla, N. D. (1985). "A unification of some software reliability models," *Siam J. Sci. Stat. Comput.,* **6,3**. 781-790.

Moranda, P. B. (1975). "*A comparison of software error-rate models,*" Texas ., Conference on Computing. 2A-6.1-2A-6.9.

Raftery, A. E. (1988). "Inference for binomial N parameter: A hierarchical Bayes approach," *Biometrica*, **75**, 223-228.

Singpurwalla, N. D. and Wilson, S. P. (1994). "*Software reliability modeling.*" International Statistical Review, 62,3 289-317.

Varian, Hal R. (1975). "A Bayesian approach to real estate assessment, in Studies in Bayesian Econometrics and Statistician Honer of Leonard

J. Savage," eds. Stephen E. Fienberg and Arnold Zellner, Amsterdam: North Holland, 195-208.

Zellner, A. (1986). "Bayesian estimation and prediction using asymmetric loss function," *J. American Statistical Assoc.* **81**, 394, 446-451

CHAPTER 5

Aue, A., Steinebach, J. (2002). A note on estimating the change-point of a gradually changing stochastic process. *Statis. Lett.,* **56,** 177-191.

Chang, Y. P. (2001). Estimation of parameters for nonhomogeneous Poisson process: software reliability with change-point model. *Comm. Statist. Simul. Comput.* **30,** 623-635.

Chen, J., Gupta, A. K. (2001). On change-point detection and estimation. *Comm.. Statist. Simul. Comput.* **30,** 665-697.

Dabye, A. S., Farinetto, C., Kutoyant, Y. A. (2003). On Bayesian estimators in misspecified change-point problems for poisson process. *Statist. Prob. Lett.,* **61**, 17-30.

Fotopoulos, S.; Jandhyala, V. (2001). Maximum likelihood estimation of a change-point for exponentially distributed random variables. *Statist. Prob. Lett.* **51**, 423-429.

Goel, A. L., Okumoto, K. (1979). Time-dependent error-detection rate model for software reliability and other performance measure. *IEEE Trans. Reliabil.* **R-28**, 206-211.

Loader, C. R. (1996). Change-point estimation using non-parametric regression. *Ann. Statist.* **24**, 1667-1678.

Muller, H. G. (1992). Change-points in nonparametric regression analysis. *Ann. Statist.* **20,** 737-761.

Musa, J. D., Iannino, A., Okumoto, K. (1987). *Software reliability: Measurement, Prediction, Application.* New York: McGraw-Hill.

Musa J. D., Okumoto, K. A. (1984). A logarithmic Poisson execution time model for software reliability measurement. Proce. 7th Intern. Conf. Software Eng. 230-238.

Nguyen, H. T., Rogers, G. S., Walker, E. A. (1984). Estimation in change-point hazard rate models. *Biometrica,* 71, 299-304.

Pham, H. (1993). Software reliability assessment: imperfect debugging and multiple failure types in software development. EG & GRAMM-10737. Idaho National Engineering Laboratory.

Pham, H. (1999). *Software Reliability.* New York: Springer-Verlag.

Pham, T. D., Nguyen, H. T. (1990). Strong consistency of the maximum likelihood estimators in the change-point hazard rate model. *Statistics* **21,** 203-216.

Pham, H., Zhang, X. (2003). NHPP software reliability and cost models with testing coverage. *Eur. J. Oper. Res.* **145,** 443-454.

Pons, O. (2002). Estimation in a Cox regression model with a change-point at an unknown time. *Statistics,* 36, 101-124.

Shyur, H. J. (2003). A stochastic software reliability model with imperfect debugging and change-point. J. *Syst. Software,* **66,** 135-141.

Singpurwalla, N. D., Wilson, S. P. (1999). *Statistical Methods in Software Engineering: Reliability and Risk.* New York: Springer-Verlag.

Xie, M. (1991). *Software Reliability Modelling. Singapore*: World Scientific Publisher.

Xie, M., Goh, T. N., Tang, Y. (2004). On changing points of mean residual life and failure rate function for some generalized Weibull distribution. Reliabil. *Engi. Syst. Safety,* 84, 293-299.

Yamada, S., Ohba, M., Osaki, S. (1983). S-shaped reliability growth modeling for software error detection. *IEEE Trans. Reliabil.* **R- 32,** 475-484.

Yao, Y. C. (1986). Maximum likelihood estimation in hazard rate models with a change-point. *Comm. Statist. Theory Method*, **15,** 2455-2466.

Zhao, M. (1993). Change-point problems in software and hardware reliability. *Comm. Statist. Theory Method*, **22,** 757-768.

Zou, F. Z. (2003). A change-point perspective on the software failure process. *Software Testing Verification Reiliabl.* **13,** 85-93.